A Comparison of Critical Chain Project Management (CCPM) Buffer Sizing Techniques

Greg Kokoskie
SYST 798 Research Project
George Mason University

19 December 2001

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (FROM - TO)
19-12-2001	Thesis	xx-xx-2001 to xx-xx-2001

4. TITLE AND SUBTITLE
A Comparison of Critical Chain Project Management (CCPM) Buffer Sizing Techniques
Unclassified

5a. CONTRACT NUMBER
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Kokoskie, Greg ;

5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME AND ADDRESS
George Mason University
xxxxx
xxxxx, xxxxxxx

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME AND ADDRESS
,

10. SPONSOR/MONITOR'S ACRONYM(S)
11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APUBLIC RELEASE
,

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Current project management literature exhibits a rise in popularity (at least in discussion) of Critical Chain Project Management (CCPM), a management concept based on Eli Goldratt?s Theory of Constraints. Some of the literature notes this to be a significant departure from current methods; others claim it is really not revolutionary at all, but is instead a coherent compilation of long-known techniques (McKay and Morton 1998). Some organizations that have implemented the technique report good progress; others report no progress. Intuitively, many of the concepts are good rules that any project should follow, regardless of whether the organization directly embraces the CCPM concept. A fair amount has been written about the differences between CCPM and traditional project management techniques (notably Steyn (2000), Leach (2000), and Herroelen and Leus (2001)). There has also been a reasonable amount written about sizing the buffers when establishing the critical chain plan (notably Goldratt (1997), Leach (1999, 2000), Herroelen and Leus (2001), Hoel and Taylor (1999)). However, while the available literature about sizing the buffers does note some general differences between the methods, it fails to mention or recommend cases where one technique is superior to another (nor does it state that such a recommendation would be inappropriate).

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	Public Release	43	Fenster, Lynn lfenster@dtic.mil
Unclassified	Unclassified	Unclassified			19b. TELEPHONE NUMBER International Area Code Area Code Telephone Number 703767-9007 DSN 427-9007

Table of Contents

Current project management literature exhibits a rise in popularity (at least in discussion) of Critical Chain Project Management (CCPM), a management concept based on Eli Goldratt's Theory of Constraints. Some of the literature notes this to be a significant departure from current methods; others claim it is really not revolutionary at all, but is instead a coherent compilation of long-known techniques (McKay and Morton 1998). Some organizations that have implemented the technique report good progress; others report no progress. Intuitively, many of the concepts are good rules that any project should follow, regardless of whether the organization directly embraces the CCPM concept. A fair amount has been written about the differences between CCPM and traditional project management techniques (notably Steyn (2000), Leach (2000), and Herroelen and Leus (2001)). There has also been a reasonable amount written about sizing the buffers when establishing the critical chain plan (notably Goldratt (1997), Leach (1999, 2000), Herroelen and Leus (2001), Hoel and Taylor (1999)). However, while the available literature about sizing the buffers does note some general differences between the methods, it fails to mention or recommend cases where one technique is superior to another (nor does it state that such a recommendation would be inappropriate).

Accordingly, practitioners of CCPM, especially those new to the concept, would benefit from a comparison between the techniques. More specifically, they would benefit from a comparison between the techniques that recommended the best buffer sizing method for their particular network.

The objective of this paper is to provide a recommendation for the best buffer sizing technique to use in a given scenario. To do that, a simple simulation will be executed on a series of sample networks and the results will be analyzed.

This paper is arranged in several sections as follows. The first section provides a brief synopsis of CCPM and discusses in a little more detail the shortcomings of previous research. The second section describes the methodology of this analysis in detail. The third section notes the assumptions and limitations of this study. The fourth section provides summary results and analysis of the simulation runs (detailed information is located in an appendix). The fifth section provides the conclusions drawn from the results. And the final section briefly outlines areas for additional research. Note that other appendices provide additional detail on other related topics. (See table of contents.)

In his 1984 book titled *The Goal*, Eli Goldratt began advocating the concept of the Theory of Constraints (TOC) for managing manufacturing operations. The theory acknowledges that every system has a constraint and that, to increase system performance, performance at the constraint must be improved. Attempting improvements elsewhere is thought wasteful because the constraint will remain the bottleneck. In the last several years, the theory has been adapted and advocated for non-manufacturing project management. Owing to its name for the largest resulting path in a project network (and the analogy that a chain is only as strong as its weakest link), it is called critical chain project management. Goldratt and others assert significant shortening of project times via this method. Indeed, Goldratt notes several significant examples where the concept has greatly reduced project times below times estimated via conventional methods. It should be noted, however, that there are also several examples where the technique was used without success.

According to Goldratt, the traditional method (i.e., Program Evaluation and Review Technique – PERT – and Critical Path Method – CPM) is seriously flawed in two primary ways: 1) it inappropriately focuses on local optimization in favor of the broader and more appropriate system optimization and 2) it provides safety time for each activity instead of for the overall project.[1]

Compared to traditional scheduling and management methods, CCPM stands out via 1) its use of median value (50:50) activity time estimates instead of those estimated via a "most likely" or higher certainty, 2) the merging of potential activity safety time into consolidated path and project buffers, and 3) the elimination of resource multi-tasking. Further, the concept advocates that the critical chain does not change once the project is started and that progress reporting can be minimized to the form of statistical process control-like checks of the consumption of the project's buffers.

Perhaps the most significant difference is the technique's use of 50:50 activity duration estimates in developing project network schedules. At first glance, the concept sounds quite unrealistic. After all, if the low-risk estimates frequently resulted in projects overrunning their proposed end date, how could simply cutting the proposed *planning* figures help the activities actually meet their schedules?

Cutting the task planning durations in half is part of the process of consolidating project safety time. Goldratt and others assert that such a consolidation requires less overall safety time than is present using the current processes (where safety time is embedded in *each* activity estimate). More importantly, the consolidation puts it where it belongs, protecting the project as a whole instead of its individual parts. The consolidation is analogous to the concept of an insurance company: for homeowners to protect their homes against loss due to fire, they could either 1) individually maintain some type of fund to cover the replacement value of their home or 2) pay a lesser amount into a central fund that would cover the cost of replacing some but not all of the homes for the group of owners. This concept is viable because it is unlikely that all the homeowners in the group would simultaneously suffer losses. The total money required in all the separate savings accounts would obviously be more than the total money required in the consolidated account.

Like the assumption that fires would not simultaneously affect all homeowners, the overruns of the activity times should not be expected to happen for all tasks: the times are 50:50 times and one should therefore expect that, for all the tasks, approximately half of them will

[1] Safety time, also called contingency time, is the difference between the high-confidence/low-risk (i.e., safe) estimated duration and the estimated average duration.

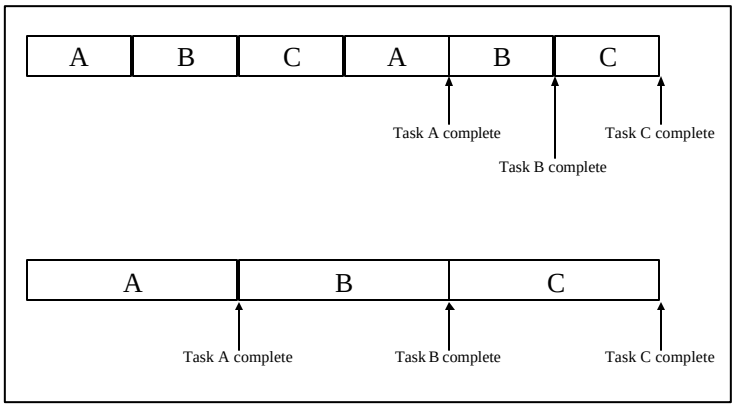
overrun their estimates and approximately half of them will underrun their estimates, helping to balance each other out.

By cutting the initial estimates for the activity durations in half, the safety time embedded in the estimates is removed. Half of it is then discarded and the other half is consolidated into buffers at the end of each network chain. The 'chain' of CCPM is differentiated from the 'path' of CPM/PERT by the explicit inclusion of resource logic. While CPM/PERT includes technological precedence logic in the networks, the concepts essentially assume unlimited resources. Accommodating resource constraints is done after the fact but the logic path remains unchanged.[1] The chain concept acknowledges that a particularly scarce resource could make tasks not on the critical path become part of the critical chain.

The elimination of multi-tasking behavior is one of the other major tenets of CCPM. The allegation is that multi-tasking is detrimental to organizations in two primary ways: 1) employees waste productivity shifting between tasks getting mentally re-acquainted with the task as well as possibly having to move to a new location or otherwise physically changing their work area, and 2) by working a little on all required tasks, the organization fails to expedite a particular one's completion.

The lost productivity having to shift focus is fairly self-evident. Especially for mentally demanding tasks, it is difficult (if not impossible) to switch from one task to another without some review of the significant aspects of the new task. This focusing period would otherwise be spent productively on a given task.

Explaining how multi-tasking fails to expedite task completion is best done with an example. The figure below (adapted from Goldratt 1997) shows two different Gantt-style work schedules for a resource on tasks A, B, and C. In the first set, the resource splits the tasks in half and works sequentially on the first half of each of the tasks and then repeats the process for the second half of the tasks. Note the resulting completion points for each task and compare them to the completion points for each task in the second set, where the resource focuses on a single task and works it to completion before beginning the next. The overall duration to complete the three-task combination is no different for the two groups, but, if each of these tasks were revenue generating, it would be far better to accomplish the second group (resulting in increased revenue earlier).



[1] The claim by some CCPM advocates that CCPM is superior to CPM/PERT because only the former explicitly treats resource contention, is unfair. While it may be true that CPM /PERT did not formally treat resource contention, reasonable managers are unlikely to ignore resource contention (except perhaps as part of the bid process in multi-project environment where it would be unknown which bids would be won, and therefore, which cross-project resources would be in contention).

The CCPM concept admonishes against sacrificing project optimality for local efficiency. In profit-based organizations, focusing on the bottom line may be good but it is important to remember to focus on the *correct* bottom line. For example, consider that an employee who is idle for a few hours per week (while still being paid his salary) may appear wasteful in the context of the particular employee, his immediate supervisor, and perhaps with the employee's coworkers. By shifting the activity schedule, it may be possible to keep the employee constantly busy. But what is really important is the *organization's* bottom line. Such changes should be made only after considering the effect on the overall project or organization.

The CCPM concept advocates that the critical chain does not change once the project is started. This is different than CPM/PERT where it is understood that if an activity on a near-critical path overruns its planned duration, the near-critical path may supplant the original critical path. In CCPM, the buffers help dampen this shifting by theoretically absorbing such overruns. Further, the concept notes that such shifts in focus are counterproductive. Owing to the uncertain nature of projects, some variation is expected.

On the concept of reporting, instead of detailed status or earned value reports, critical chain reporting consists of statistical process control-like checks of expected buffer consumption. Because of this focus on buffer consumption as the primary means of monitoring the project, the technique is sometimes also called critical chain/buffer management. Estimates to complete for each task are developed and then summed for each chain and compared to the plan to determine whether and how much of the buffer is expected to be consumed. The table below shows the recommended actions for various buffer penetration amounts. Other simple rules are also sometimes employed to trigger action (e.g., three consecutive reports of buffer consumption increases).

Buffer Consumed	Recommended Action
< 1/3	None; within statistical control
1/3 to 2/3	Develop action plan
> 2/3	Implement action plan

Table 1: Buffer Penetration Thresholds and Recommended Actions

As noted earlier, CCPM grew out of Goldratt's Theory of Constraints. Goldratt publicized the concept primarily through his 1997 book, *Critical Chain*. While the book is an enjoyable easy read, it presents the general concepts but does not provide detailed implementation guidance. Other sources (Leach (1999, 2000), Newbold (1998), Steyn (2000) provide much greater detail, providing the practitioner with better understanding of how to execute CCPM.

Because the technique is relatively new, access to empirical evidence is limited. Accordingly, comparisons that include CCPM are generally done via simulation. Most of these comparisons have focused on comparison of management methods (e.g., traditional management versus CCPM). Herroelen and Leus (2001) conducted a detailed experiment, effecting such a comparison. While they do describe the several popular buffer sizing techniques, they include only two (the "half" and root square error methods) in their experiment. They note that the "half" method generally results in larger project buffer sizes but do not attempt to correlate one buffer sizing technique to a network with a given set of characteristics. Cook (1994), while predating the CCPM movement, used simulation to compare traditional, just-in-time, and theory

of constraints techniques, but he limited his discussion to a manufacturing environment and did not discuss buffer types or make any recommendations for the best technique for a given situation. Hoel and Taylor (1999) used simulation in their investigation of buffer sizing but also did not correlate a buffer sizing technique to a network with a given set of characteristics.

All these works have one central shortcoming from a practitioner's perspective. While they certainly shed light on various aspects of traditional versus CCPM management, they do not provide the practitioner a recommended technique. They similarly do not state or attempt to prove that any one technique is best for a given scenario. While it would be ideal to provide practitioners with a recommendation, it would also be valuable to note (if appropriate) that there is no significant performance benefit of one sizing method over another or if there is no correlation between readily determinable network characteristics and a preferred sizing method.

The primary buffer sizing methods are summarized in the table below. Goldratt (1997) advocated a very simple method, sizing the buffers at half the amount that was trimmed out in reducing the activity duration estimates from their high confidence value to the 50:50 mark. Leach and others acknowledge Goldratt's method as reasonable and offer an alternative that uses the root square error (square root of the sum of squares of the amount of time trimmed from each activity). Hoel and Taylor advocate simply converting the tasks on the feeding chains to early start tasks (thereby sizing the feeding buffers at the amount of slack on the non-critical chains) and sizing the project buffer via Monte Carlo simulation. The method that Newbold (1998) attributes to Rizzo is refuted by Herroelen and Leus (2001) as containing a mathematical error. Herroelen and Leus (2001) offer a correction to the method but it presumes a lognormal activity distribution. The other methods do not levy such a requirement.

Method	Primary source	Description		Comments
		Feeding buffer	**Project buffer**	
Half	Goldratt	½ of time trimmed out of path when reducing from high confidence estimate to 50:50 estimate	Same as for feed buffer but applies only to critical chain	
Root Square Error	Leach	$[(S-A)_1^2 + (S-A)_2^2 + \ldots + (S-A)_n^2]^{1/2}$ for all n tasks on the path, where S = high confidence (safe) estimate A = 50:50 (average) estimate	Same as for feed buffer but applies only to critical chain	Also noted in Herroelen and Leus
Monte Carlo and Slack	Hoel and Taylor	None; use early start schedule for non-critical chain tasks; resultant slack becomes the feeding buffer	Perform Monte Carlo analysis on project network; project buffer = difference between 90th percentile and 50th percentile values	
Two Standard Deviation	Newbold	$2 \times \{[(S-A)_1/2]^2 + [(S-A)_2/2]^2 + \ldots + [(S-A)_n/2]^2\}^{1/2}$ for all n tasks on the path, where S = high confidence (safe) estimate A = 50:50 (average) estimate	Same as for feed buffer but applies only to critical chain	Requires lognormal activity duration estimates
Modified Two Standard Deviation	Herroelen and Leus	2*StdDev of chain (calculate StdDev via min, med and max estimates for each task; apply correction factor if small number of tasks)	Same as for feed buffer but applies only to critical chain	Requires lognormal activity duration estimates

An interesting underlying philosophy of CCPM includes an attitude of "good enough." The method recognizes the high level of uncertainty in projects and specifically advises against attempting to provide too much detail in schedules. A highly detailed schedule requires not only a large amount of effort to establish it in the first place but also to maintain it. Even the smallest changes can have far-reaching effects, potentially forcing managers to spend time *correcting* the schedule instead of *managing* it. Also, the method recognizes that there is no perfect algorithm for resource leveling. Accordingly, detailed attempts to identify an optimal schedule are misplaced; one should not agonize too much when establishing the plan. Similarly, identifying and attempting to control every resource for every activity is also unnecessarily difficult. Leach (2000) advocates focusing only on the *primary* resource for each task. (If this is not possible or desirable, he adds, it may be necessary to decompose the task into its subordinate elements.)

Introduction

This paper attempts to identify and characterize the differences between the performance of the different buffer sizing methods. Accordingly, the root question is: are there differences? Presuming that differences do indeed exist, the next logical question is: what is the nature of the differences? Specifically, is one method better than the other in all cases or only in certain situations? If the "best" process is different for different situations, which situations correspond to which "best" process? And perhaps most importantly, is there a simple measure to characterize the proposed network, allowing the manager a practical way of identifying the recommended buffer sizing method for his current situation?

In general, the experiment was conducted in several steps. First, the different sizing methods were used to develop critical chain plans across a set of well-known problems. The plans were then entered into a simple simulation and their performance was compared. The comparison considered likely groupings of performance by network complexity and level of resource constraints.

This section is subdivided into parts as follows: Problem selection, Network characteristics, Input distribution, Performance measurement, Input distribution, Establishing the critical chain plan, Running the simulation, Analyzing the results. Each part describes the particular aspect of the methodology in greater detail.

Problem selection

The resource constrained project scheduling problem is a common topic in operations research and management science literature. Various heuristics and solution methods have been developed in an attempt to efficiently identify a project's optimal schedule.

Problems in the literature have a wide range of complexity. On one end of the range, the problems are fairly simple, having only a few activities and a relatively small number of precedence relations and a requirement for only a single resource for each activity. On the other end of the range, the problems are quite complex, having a large number of activities that are highly inter-connected and that have multiple resources required for each activity. In addition, the activities in these complex problems may have multiple modes such that the specific resource requirements for the activity differ depending upon the mode in which the activity is executed. Complex problems may also contain contingency paths such that the timing or results of an activity drives the duration, resource requirements, or precedence relations of other activities.

The selection of problems is important to ensure an adequately diverse group in order to test various solution methods. One of the relatively common datasets stems from Patterson (1984), a work frequently cited in the literature. In the work, Patterson tested three enumerative network solution approaches across a fairly exhaustive set of 110 problems that were relatively known at the time. Since then, tools have been developed to generate network problems, with the intent that they could be used systematically to develop an adequately diverse group. The most notable of the tools and the one that appears to be the standard is called ProGen (short for Project Generator).

A library of ProGen-developed networks was developed and compiled by Kolisch and Sprecher and described in their (1996) article. The entire problem library currently consists of a vast number of problems and is available via the internet. The library contains primarily multi-mode resource problems but does contain some single-mode resource problems. Of interest, the library also contains the Patterson problems and those generated by several others.

In selecting the problems for this work, I intended to use a set that was commonly acknowledged by researchers as being representative. Initial literature searches implied that the Patterson set met such requirements as it was used in other works (e.g., Herroelen and Leus (2001)). However, Kolisch, Sprecher and Drexl (1995) cite weaknesses with this and several other datasets. Specifically, they state that 1) the problems were not generated via a controlled design of parameters (thereby calling into question whether the sets are adequately exhaustive) 2) only the single-mode case and makespan minimization is considered (thereby failing to accommodate for other complexities of the real world) and 3) that the set is more easily solvable than other sets with today's computers and should therefore not be considered as a benchmark (a comparison of solution methods is usually conducted via a comparison of computer solution times).

These weaknesses would be important to counteract when developing a potential solution method for the resource constrained project scheduling problem. It would be important to know whether the solution holds for all problem types or whether it is instead viable only for a specific grouping. However, the weaknesses are less important in simply observing the behavior of a process upon a sample of problems than when one is attempting to identify a solution method that is intended to apply across all problem types.

Lack of an exhaustive set is not a significant obstacle because this paper simply intends to make generalizations about the relationship between the various buffer sizing methods and subsequent performance of the network. Such generalizations would apply provided the problem set was reasonably widely representative even though it may not be exhaustive. The single mode and makespan minimization limitations are similarly not obstacles for the same reason. This model is an attempt to simplify the real world and draw potential conclusions, not perfectly replicate it. One need only recognize these limitations and not attempt to develop conclusions far outside the range of problems considered. The third shortcoming is not relevant to this experiment because there is no need to base measurements on computer solution time.

The Patterson dataset was selected because it is a reasonably broad, classical set that has been used by a number of other works. Despite the limitations stated by Kolisch, Sprecher and Drexl (1995), the set remains viable as a benchmark to use in making comparisons. While it may not be an exhaustive set of all project types, it is nonetheless reasonable, especially as a "first look". Subsequent efforts could broaden this investigation to other problems.

It should also be noted that the entire Patterson dataset was not used. Instead, the problems were ordered randomly and the implementation was then initiated following this random order. In this way, since time was recognized as a potential factor in completing the analysis, I could end the analysis once a sufficiently large set of problems were investigated and still be reasonably confident that the results would not change if the entire dataset was used. In the end, a total of eight problems for each buffer sizing method were evaluated.

Randomization is critical for assigning problems to the experimental units. The typical method would be to establish the pool of available problems that meet the requirements for each category then select the desired quantity at random from the pool. However, because there was inadequate time to develop summary statistics for the entire Patterson dataset, an alternative approach was used. Problems were placed in random order and then inspected in this order to identify the first one to meet the requirements of the given experimental unit.

Network Characteristics.

Initially, it seemed valuable to assess whether a given buffering process would work better for various *types* of projects (e.g., research and development, product development, construction, software development, etc.). While such a categorical assessment would undoubtedly be valuable, it could likely turn out to be impossible because it would be difficult to identify the perfect paradigmatic problem for each category. Each of the categories varies from the relatively simple to the complex, the small to the large, the loosely constrained to the tightly constrained, etc.

Accordingly, it was decided that it would be better to simply characterize a sampling of networks. That way, results could be applied by anyone in any of these genres: one could simply calculate the appropriate characteristic values for a given network (regardless of whether one is working in aerospace, software development, etc.) and then apply the buffer sizing technique that is best for the characteristics.

It was somewhat challenging to determine a reasonable set of characteristic measures. While some immediately came to mind (e.g., network size, number of precedence constraints, number of resources, etc.), they were neither suitable nor sufficient. For example, network size was determined to be inapplicable because in many cases a network could be considered either as a subnetwork of a larger problem or to possess its own internal subnetworks. In either case, depending upon one's perspective, one may have a "different" problem. The intent was to provide an assessment of the buffer sizing processes in a truly practical environment. Accordingly, even if network size would have an effect, it would not be practical.

Across the measures, an attempt was made to preserve the notation of Kolisch, Sprecher, and Drexl (1995) and Kolisch and Sprecher (1996). Their notation was used as much as possible for uniformity. When necessary, new variable names were created such that they might fit into the naming convention as much as possible. Note, for example, the uncertainty measure below. The name d^{unc} was used because of the similar duration measures, d^{min} and d^{max} and in order to avoid confusion with U^{min} or U^{max}.

It seemed appropriate to include measures that related to the network's complexity, and the level of resource contention. A search of the recent literature in this area resulted in several works in which network characterization was used. Besides the two works in the preceding paragraph, Brucker, Drexl, Mohring, Neumann, and Pesch (1999) and Elmaghraby (2000) had some relevance, although the notation in the Kolisch et al. works appeared more likely to be widely accepted (due at least in part to their extensive and accessible problem library). Other works were too general to be of assistance. The focus of the characterization by Kolisch et al., however, was upon describing a network that would be developed via an automated tool (ProGen). As such, while the measures were not inapplicable, they were less usable from a practitioner's viewpoint.

It seemed important to include a measure of network complexity or connectedness. Possible metrics could include 1) the number of feeding (i.e., non-critical) paths, 2) the percentage of tasks on the critical chain or path (such that a higher percentage would imply lower connectedness because the network would be more serial than parallel), or 3) the ratio of the number of paths to the number of tasks. In keeping with Kolisch, Sprecher and Drexl (1995) and their predecessors (they cite Pascoe (1966) and Davis (1975)), the network complexity, NC is shown below. It equates to an average number of non-redundant arcs per node. Note that a perfectly serial network would have J-1 arcs. Note also that redundant arcs are not counted;

$$RF = \frac{1}{J\text{-}2}\frac{1}{|R|}\sum_{j=2}^{J}\sum_{r\in R}\begin{cases}1 & \text{if } k_{jr}>0,\\ 0 & \text{otherwise.}\end{cases}$$

while including them certainly makes the network *appear* complex, it would not necessarily *be* complex.

$$NC = \frac{\text{Total \# non-redundant arcs}}{J}$$

Kolisch, Sprecher and Drexl (1995) note that Pascoe (1966) introduced a measure called resource factor that was also subsequently used by other authors. They adapted the measure to include both renewable and non-renewable resources and to account for multiple modes of the activities. They acknowledge that their measure reduces to Pascoe's measure for the single-mode case for problems with no non-renewable resources. Since the problems used in this analysis are only single-mode activities requiring only renewable resources, it is easier to use the less complex Pascoe measure. The measure is shown below with one minor modification from Pascoe's original measure to prevent the dummy super-source and super-sink from affecting the resultant value. Essentially, the measure is a density of the non-zero elements of the *activity ´ resource* array. Its range is [0,1] with 0 corresponding to the case where no activity requires any resources and 1 corresponding to the case where every activity requires some non-zero amount of each resource. Note that this does not differentiate between networks whose *activity ´ resource* array is fully populated by low non-zero numbers and one whose array is fully populated by high non-zero numbers.

Kolisch, Sprecher and Drexl (1995) note that Cooper (1976) introduced a measure called resource strength that was subsequently used by other authors. They criticize the measure on three counts: 1) it is not normalized to the interval [0,1]; 2) that, when generating problems, a small resource strength does not guarantee a feasible solution; and 3) that the measure does not differentiate between networks that are predominately parallel and those that are predominately serial. (In parallel networks, there is a greater potential for simultaneous demand on a given resource that is not present in serial networks.) They propose a modification to the measure that also generalizes it to the multi-mode, multi-resource type case. However, the added complexity does not seem to be worth the payoff, especially for this study. Accordingly, this study uses Cooper's measure as stated below. (As with the resource factor, this measure has one minor modification from Cooper's original measure to prevent the dummy super-source and super-sink from affecting the resultant value.)

$$RS_r = \frac{K_r}{\frac{1}{J\text{-}2}\sum_{j=2}^{J-1} k_{jr}}$$

However, because the measure is a ratio of the resource availability to the *average* demand, no specific conclusions can be drawn from it. It can give an indication of the strength level of the resource in general across the entire problem but it cannot be used to determine task feasibility. (A few activities could require more of a resource than is available (i.e., $k_{jr}>K_r$). But, if there are a number of other activities in the network with lesser resource requirements, the

$$d^{unc} = \frac{\sum_{J:d>0} d_j - \bar{d}_j}{J:d_j>0}$$

average demand could be below the maximum available amount of the resource (i.e., Avg $k_{jr}<K_r$,) creating the appearance that adequate resources are available.) Nonetheless, the measure still holds promise in a general, overarching way.

To overcome the most significant limitation of Cooper's resource strength measure (as noted by Kolisch, Sprecher and Drexl above), additional measures of resource strength were also

$$RS3_r = \frac{K_r}{(\sum_{j=1}^{J} d_j k_{jr}) / \sum_{j=1}^{J} d_j}$$

collected. The resource strength measure number two, designated $RS2_r$, measures the proportion of the resources available to the minimum level at which they would need to be set to prevent resource contention. Accordingly, values of $RS2_r \geq 1$ imply that the resource is not constraining the schedule; values <1 imply that the resource is constraining the schedule. (It should be noted that $K_r^{reqd:LF}$ is a value based on the network using the estimated durations as deterministic values.) Resource strength measure number three, designated $RS3_r$, is a modification to Cooper's measure, changing it from a simple average to a weighted average based on the activity's estimated duration. Because it is an average, it is still subject to the same limitations as RS_r.

It also seemed important to investigate the effects of uncertainty of the duration estimates, acknowledging in some cases it is very difficult to obtain reasonably close tolerances on the initial task estimate durations. The measure, denoted d^{unc}, is shown below. Note that only activities with non-zero durations are included.

The last measure considered was average task duration. It was suspected that CCPM's general rules of dealing with fractional time periods in reducing estimates to their 50:50 values and in sizing the buffers (i.e., rounding up to integer values) could have impact, especially in networks where the average task duration is low. In such networks, rounding up to integer values provides a higher percentage of built-in safety time on a given chain explicitly in the buffer as well as implicitly within the (slightly higher) 50:50 duration estimate. The two examples below illustrate the impact via two 2-task chains. Although the "half" buffer sizing method is shown, the concept applies across all buffer sizing methods (if for no other reason than their initial schedules are laid out based on reduced activity durations that are rounded up to integer values).

Task ID	High confidence task estimate	Actual "half" amount	"Half" rounded up to integer	Buffer size	
Example 1: Low, odd activity duration estimates					
A	3	1.5	2	1	
B	1	.5	1	1	
Total	4	2	3	2 -> buffer= 2/4 = 1/2 of original chain duration	

Example 2: Higher, even activity duration estimates

C	4	2	(2)	1
D	8	4	(4)	2
Total	12	6		3 -> buffer = 3/12 = 1/4 of original chain duration

Performance measurement

At first reflection, it might be thought that an appropriate performance measure would be whether the plan met or failed to meet the planned delivery date. After all, one of the primary goals of a project scheduling mechanism is to meet an established delivery date. On-time delivery is one of the primary objectives of any project organization. However, establishing the performance of the developed critical chain schedule requires some extra consideration. In general, one would expect that, if the buffers were sized adequately, the project would meet its stated delivery/completion date. Accordingly, schedules that completed their project within the planned project buffer would be viewed as superior to those that overran their project buffer (i.e., were completed after the planned delivery date). By extension, it might be reasonable to assess that buffer sizing techniques that consumed less of the project buffer to be better than those techniques that consumed more of the project buffer.

However, consuming a lesser amount of the project buffer is not necessarily always good. While it is apparent that overrunning the project's planned end date would be bad, one must also consider the ramifications of significantly underrunning the project's planned end date. Large amounts of safety time can be detrimental. It is important to be able to deliver a product (complete a project) by the established delivery date. A demonstrated history of such performance allows potential customers to have faith that the company will make good on its promises. Delivery dates that are proposed to be far in the future may likely continue to receive potential customer good will (due to expected ability to meet promises). However, excessively distant promised dates run the risk of being perceived as unresponsive by customers. For example, if several firms are bidding on a project and one has an earlier delivery date than the others, it may well win the contract (assuming the customer believes he will get quality performance at a reasonable price and that the vendor will indeed deliver by the promised date). There may be cases where large safety time amounts are good – e.g. on projects deemed as "high risk." Nonetheless, for a given "high risk" project, the vendor that promises an earlier delivery date is likely to win the contract over other vendors that promise a later date (again, assuming quality performance, a reasonable price, etc.).

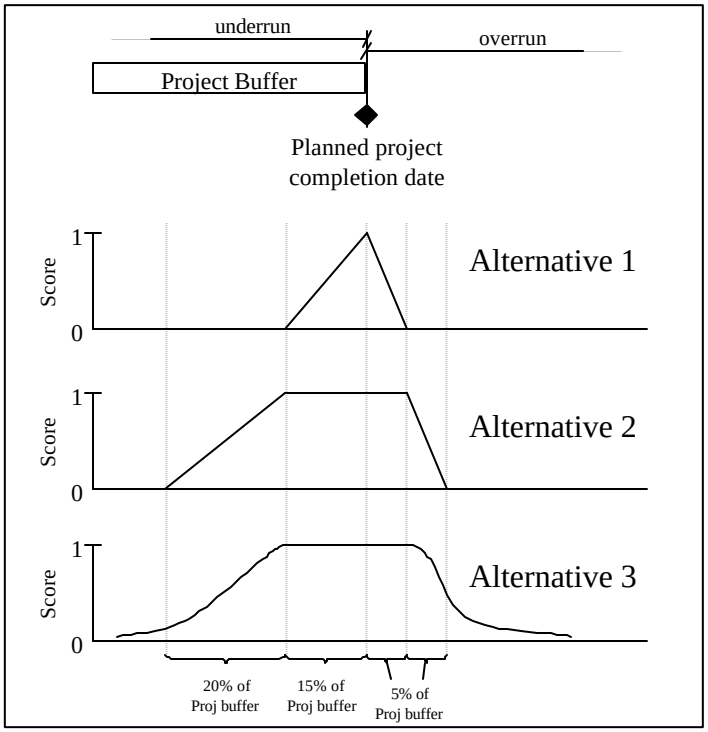
Accordingly, the size of the project buffer can be viewed as a trade-off. On one hand, it is important to meet promised delivery dates, for failing to do so can result in financial penalties, loss of potential customer confidence, and ultimately, loss of future business. On the other hand, it is also important not to build in excessive safety time in the project schedule because doing so could also lead to loss of future business as customers migrate to other vendors that promise (and meet) earlier delivery dates.

The trick is establishing what to consider "excessive." Before delving into the proposed amount, one must realize that this simulation experiment does not include the effect of management action. (See assumptions and limitation section.) One would expect competent management action to shrink the simulated size of the project by some amount. (This, of course, assumes that the management team recognizes an impending buffer penetration, that the penetration occurs early enough in the project to allow the team to have an effect, that crashing

the network is possible, and that their actions have the desired effect.) By not including effects of management action, the simulated project is essentially running on autopilot. Once the plan is established, the management team takes no further action. The management team, if it were present in the simulation, should have a positive effect when reacting to potential overruns. Since the team is not present in the simulation, it would be reasonable to assess as "good" those simulated projects that slightly overrun their planned delivery dates (i.e., the expectation is that if they were present, they could fix small overruns).

It is interesting to note that Williams (1999), in describing ways of enhancing realism in simulations, notes that management action is one of the hardest elements to simulate. He adds that the difficulty lies not in implementing an algorithm or process in the simulation but rather in developing actual expected effects (i.e., detailing what effects the algorithm or process would actually have).

Accordingly, for a simulated project with no management action, ideal performance may slightly overrun the planned completion date. Resolving how to score the performance remains the issue. Is 10% overrun too much? Is 5% underrun as good as 5% overrun? The diagrams below show several of the options considered.



Alternative 1 allows a perfect score if the project ends exactly in accordance with the plan. Scores taper to zero for small underruns and, more sharply, for small overruns. A variant of this alternative would be to vary the amounts shown to other values, e.g., -25%,+15%. This scoring method is quite simple and, because it has one perfect score that is located at the planned completion date, it is also intuitive.

Alternative 2 allows a perfect score if the project ends exactly in accordance with the plan, plus or minus a small amount. Beyond these points, the score tapers to zero. Although this scoring method is slightly more complex than Alternative 1, it is also quite simple. Because the perfect score is allowed to be a range that straddles the planned completion date, it requires the

reader to acknowledge that there is little difference between the points along the flat top of the curve, and that because of the aforementioned lack of management action in the simulation, that small overruns are as worthy as small underruns.

Alternative 3 is similar to Alternative 2 but accommodates a wider range of scores. In the previous alternatives, no differentiation is made between a project that ends prior to its planned completion date by 40% of the project buffer amount and one that ends prior to its planned completion date by 60% of the project buffer amount. This scoring method is the most complex of the three, requiring a more complex formula for the tails. Nonetheless, it is also the most realistic.

For simplicity, Alternative 2 was selected. Buffer consumption percentage was collected as part of the simulation. The scoring metric was applied for each run of the simulation and the results were analyzed.

Input Distribution

There seems to be a general consensus that most project activity distributions are right-skewed (i.e., with a longer right tail). In fact, most sources assume that the reader agrees with such an assertion, effectively treating the assertion as a foregone conclusion. A few sources, such as Gutierrez and Kouvelis (1991) actually take steps to explain the underlying reasoning for such a distribution and its implications. There are also cases for symmetric distributions (e.g., for relatively simple processes that have been done before – the shape would likely become closer to that of a manufacturing process). Such activities are certainly not the predominant type in projects, where, by definition, the undertaking is relatively new or unique. There is even an example for left-skewed distributions (e.g., for activities with extremely firm deadlines – one might work on the activity as long as possible, improving its quality beyond some initial amount with the distribution of activity completion dates concentrated shortly before the deadline).

Nonetheless, a right-skewed distribution is the generally accepted form. However, different authors advocate different specific shapes, with the primary three being triangular, beta and lognormal.

Wendling and Lorance (1999) recommend the use of a modified triangular distribution in simulations due to the difficulty in developing the quality of the inputs required for most distribution types. They note that the beta and lognormal distribution types are generally superior however, and recommend their use if the parameters can be reasonably determined.

A triangular distribution is desirable for simulation because it is conceptually easy – both to develop estimates as well as to implement. Its straight line shape and closed form makes calculations simple and eliminates consideration of long tails. It is easy to communicate to others and can be shaped in various ways: symmetrical or skewed to the right or left.

The beta distribution has been frequently used in activity time estimates and remains popular as a result of its initial association with the PERT technique. While some authors decry its use, others such as Kamburowski (1997) defend its legitimacy. It has a realistic (smooth) shape and its parameters can be varied to match differently shaped activity time estimates. However, it can be conceptually difficult, because its shape can vary widely, even with small changes in its shape parameters. It can be difficult to elicit estimates of its descriptive parameters, but given optimistic, pessimistic and most likely time estimates, its parameters can be estimated via the method shown in Badiru (1991). Its lack of closed form accommodates reality, allowing for non-zero probabilities over a potentially infinite range of values.

A lognormal distribution also has a more realistic (smooth) shape than the triangular distribution and is also frequently used in representing activity duration estimates. However, besides some potential difficulty meeting the assumptions for normality, the simulation tool did not conveniently handle the transform necessary to allow the distribution to begin at values other than zero.

The simulation tool used did not conveniently handle beta distributions that started at non-zero values. Instead, it required the minimum value to be zero. While a subsequent transform could be applied, it required 1) generating additional input values and 2) would not conveniently be included in the automated reports feature.

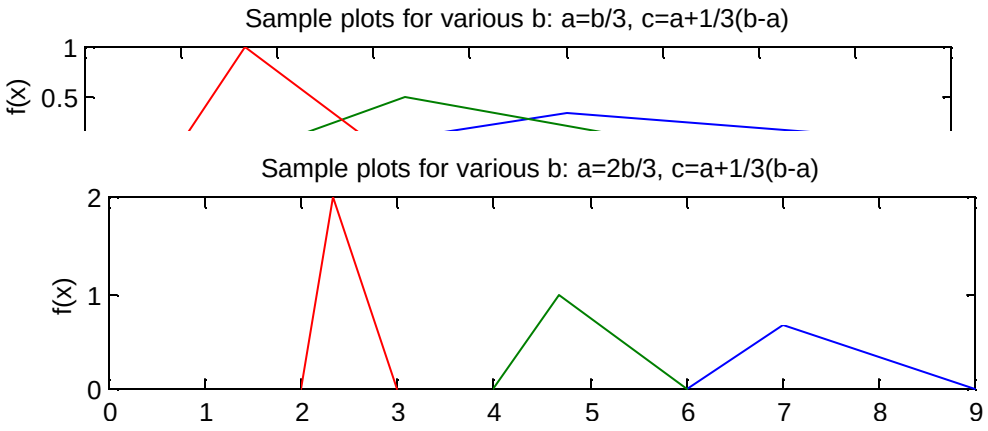
Because of its simplicity and its reasonability, the triangular distribution was chosen as the input distribution of the activity durations. The lack of smoothness of the shape of the distribution is not expected to make a significant difference. In any case, it would probably be far overshadowed by the variability of the particular distribution (whichever one is chosen).

Since the sample problems provided only one duration for each activity, the other associated parameter estimates had to be generated. The activity duration value in the problem (d_j) was used as the triangular distribution's high confidence "b" value; the low confidence "a" and most likely "c" values were then calculated based on the "b" value.

Decimal values were allowed in describing "a" and "c" for shaping the input distribution ("b", as given in the problem statement was an integer value). While duration estimates elicited from functional experts would likely be integer values (other than perhaps for short duration tasks), the decimal values were necessary for the subsequent assessment of the impact of task variability. If calculated values were rounded to integer values for the input distribution, variability would not be adequately controlled, making any subsequent conclusions suspect.

In choosing the "a" and "c" values, I considered establishing a mathematical rule (e.g., a=1/3*b) to develop the missing values. In doing so, I further considered allowing the proportion to vary randomly or to fix it as a constant. While the randomization would perhaps result in the most realistic scenario, I chose to use constant values throughout the problem generation process. Doing so would allow an investigation of the effect of task estimate variability on the buffer performance. (This could be accomplished by executing one iteration of the problem set with "a" and "c" set at one value and then repeating but with "a" and "c" at different values.)

The high and low estimate variability values for "a" were chosen as 1/3*b and 2/3*b, respectively. The value of "c" was chosen as 1/3*(b-a) for both the high and low estimate variability runs. The figures below show the resultant shapes of the two general processes. They each show plots of the triangular distribution for sample durations of $b \in \{3,6,9\}$. The first figure represents the high estimate variability case (with a=1/3*b) and the second figure represents the low estimate variability case (with a=2/3*b).



Initially I was concerned about the way the estimates vary with respect to the magnitude of the provided duration (i.e., the "b" value of the problem description). However, further reflection convinced me to consider this effect wholly reasonable – it seems likely that the variance for an activity that is expected to take a long time would be greater than (i.e., consist of more time periods) the variance for an activity that is expected to be completed in a short time. In other words, it is perfectly reasonable to expect greater precision in estimating a simple task than a long complex one. (Of course, one can always find exceptions. Some long tasks, such as regulatory waiting periods, could have relatively small variances. Nonetheless, in general, the expectation that greater variance exists in longer tasks is reasonable.)

Establishing the Critical Chain Plan

Developing the critical chain schedule is no simple task. Herroelen, Leus, and Demeulemeester (2001) make this point and note how at least two different critical chain software tools (one that has an integrated critical chain component and one that has a critical chain component as an add-in to the basic program) result in different schedules. Goldratt (1997) and Leach (2000) describe a manual method in which one moves pieces of paper representing the tasks around on a table or board in order to establish the plan and ensure resource and precedence logic constraints are met. However, it becomes exceedingly difficult to keep track of both resource and precedence requirements in moderate to large networks. This is especially true for multi-resource tasks or networks with high resource contention. This may be why Leach (2000) advocates narrowing the focus of the project's activities until there is only one significant resource per activity. Doing so can avoid the involved logic checking that results when each activity has the same resources assigned (albeit in different combinations than the other activities).

Herroelen, Leus, and Demeulemeester (2001) point out several problem areas that can occur when establishing a critical chain schedule. They note that these areas could cause problems because much of the literature does not discuss in detail the intricacies involved, leaving practitioners without readily accessible adequately detailed knowledge to forestall the problems. They develop four different baseline schedules for the same simple network and point out flaws in Goldratt's and Leach's assertions about the choice and ambiguity of the critical chain. However, I believe their attacks are not warranted. Goldratt's assertion that the choice of critical chain does not matter is somewhat out of context; I believe the assertion was based on chains of equal durations within a single plan rather than ones of differing lengths due to different development methods. The authors similarly do not note that Leach provides an explicit step to examine the baseline schedule to identify modifications that could shorten the overall project duration.

To develop the critical chain plan from the original problem information, the following steps were taken. The steps correspond to general concepts in Goldratt (1997) and the more detailed implementation descriptions of Leach (1999, 2000) and Newbold (1998).

First, a late finish schedule was developed using the reduced activity durations. The schedule was based solely on precedence relations; resources were initially allowed to be over-allocated. Microsoft Project was used as the primary tool. This schedule was then modified to meet resource constraints by starting at the end and working backward, resolving conflicts by inserting tasks in priority order (i.e., higher priority tasks inserted first would be later in the project than contentious tasks of lower priority). Priority was determined based on critical path

durations for the network segments up to and including the tasks in conflict – the task with the longest path preceding it received higher priority.

The critical chain was identified as the longest continuous path through the load-leveled network. The load-leveled network was briefly reviewed to identify obvious ways of re-sequencing the tasks to shorten the overall project duration.

Buffers were sized in accordance with the corresponding method and inserted into the plan. As expected, the most significant difficulties occurred when inserting buffers on the feeding chains. Any new resource contentions that developed due to the insertion of the feeding buffers were resolved. Newbold (1998) emphasizes that the calculated buffer sizes are guidelines only; they can and should be adjusted based on a variety of factors (such as confidence in resource availability, number of tasks on the critical chain, number of significant risks, the importance of delivery time, etc.). Nonetheless, since the objective of this paper was to examine differences in the sizing methods, the buffer sizes were not modified to attempt to accommodate any of these factors. Buffer sizes were modified, however, to resolve gaps that would otherwise be forced into the critical chain. Such gaps can occur when feeding buffers are inserted on a feeding chain that includes predecessor tasks from the critical chain. The gaps were resolved per Leach's (2000) instructions: for small gaps, the feeding buffer was reduced to eliminate the gap and the amount that was removed from the feeding buffer was added to just prior to the project buffer. Such action does not change the overall length of the project: the critical chain with the gap removed and the increased buffer at the end of the project is the same as the critical chain with the gap and the original project buffer.

For all buffer sizing methods, the duration stated in the problem (d_j) was assumed to be the "safe" estimate (sometimes denoted "S" in the literature). This high confidence value (assumed to correspond to ~95% confidence) was also used as the "b" value in the triangular distribution. For the "half" method of buffer sizing, this "S" value was simply divided by two to determine the 50:50 time. The resulting reduced activity durations (and the corresponding buffers) were rounded up to integer values. For the "RSE" method of buffer sizing, the "most likely" "c" value was assessed as the Average time estimate (sometimes denoted "A" in the literature). Decimal values were allowed for this value throughout calculations but final buffer sizes were rounded up to integer values.

In cases of precedence networks that have multiple tasks whose only successor is the dummy super-sink, the feeding buffers for the non-critical chains were allowed to run concurrently with the project buffer. This concept does not violate any of the tenets of CCPM but none of the implementation literature I could find specifically describes this nuance. In the example below, the first row of tasks is the critical chain and the second row is the feeding chain. Both tasks *y* and *n* have only the dummy super-sink as a successor task. Thus, the project is complete when both *y* and *n* are complete. The normal convention is to have the feeding buffer "completed" prior to the next task on the critical chain. Remember, however, that feeding buffers are established to prevent feeding chains from delaying the critical chain. In this case, since there are no additional activities on the critical chain, the feeding chain need not "merge" prior to the project buffer: provided that the feeding buffer is less than the project buffer, it can be scheduled concurrently with the project buffer (only contingency work would be occurring during either of the buffers' scheduled durations and it should be completed by the end of the buffer).

Calculations in the simulation for consumption of project buffer accounted for the potential case where task *n* could be completed later than task *y* and the actual project completion date (i.e., the later of the completion of task *y* or *n*) was used in determining the percentage of project buffer consumed.

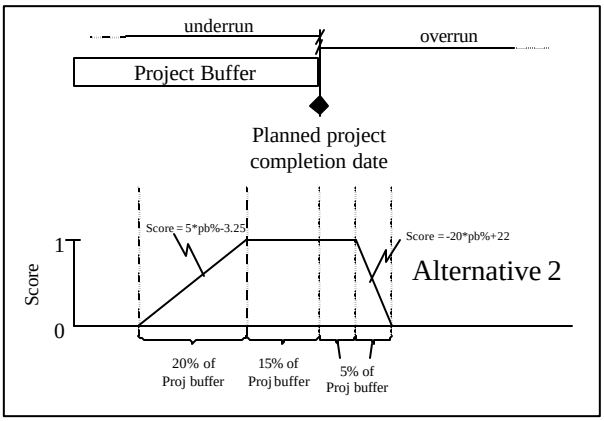
Running the simulation

The plan was then entered into the simulation. The simulation was a simple spreadsheet using Microsoft Excel as its basis with Decisioneering's Crystal Ball add-in to provide the Monte Carlo simulation and the automated reporting formats. (See description of tools in Appendix E for information on Crystal Ball.) It should be noted that the most difficult part of implementing the simulation was ensuring all constraints (both logical precedence relations as well as resource-derived precedence relations) were included in the examination of the total project duration. Because the simulation was not a time-step simulation, the final checks of the project duration included in some cases relatively long "IF" statements). (See assumptions and limitations section.)

Thus, the simulation performs a consolidated "what happened" type of view, looking collectively at the results of each task. An alternative would be to use a classical simulation tool such as Rockwell Software's Arena that would be time-phased, stepping through the project lifespan and accomplishing each task in small increments. Doing such a simulation would be more representative of how a project would progress (and would make simulated management action more realistic if it were desired to be implemented). However, the method used is reasonable in light of the scope of this investigation.

The simulation was run 500 iterations for each problem-buffer sizing technique combination. A report was created for each of the feeding buffers, the project buffer and project duration forecast values. Reports for each run were conveniently generated by Crystal Ball and include histograms of data, summary statistics and icosatiles. Also included in each report was a record of the parameters for each activity duration ("assumptions" in Crystal Ball). Data for each of the runs for the following parameters was also extracted: project buffer consumed, project buffer overrun, percent project buffer consumed, and simulated project duration. Printouts of these reports are located in Appendix E.

Each run was scored via the second of the previously-presented alternatives (shown again below) and the scores were then compared.



Analyzing the results

Designing the experiment was more time consuming than initially expected. Netel, Wasserman and Kutner (1990) proved invaluable in this respect and served as the basis for sample sizing, experimental design, etc.

The objective of the paper is to examine the effects of the buffer sizing techniques. It seemed apparent from the outset that a single-factor design was appropriate such that the different buffer sizing techniques represented the different factor levels. It also seemed clear that a repeated measures ANOVA design was appropriate, with each buffer sizing method applied to each of the selected problems. Repeated measure designs have the primary advantage of precise comparisons because sources of variability between subjects are excluded from experimental error. The primary disadvantages of repeated measures design, interference, was not present due to the nature of the problems. The "dumb" problems assured that there would be no interference effects across the buffer sizing techniques for each of the sample problems (e.g., order of the treatment – application of the buffer sizing technique – was irrelevant).

However, since no single model seemed perfectly appropriate, it appeared that some adaptation of one of the basic models would be necessary. The diagram below shows three primary approaches that were considered.

Approach 1

	Experimental units			
Factor	*j*=1	*j*=2	*j*=3	…
i=1 "half"	a	b	c	…
i=2 RSE	a	b	c	…

a,b,c,… = problems selected at random from Patterson dataset

Approach 2

Block	Experimental units
1	(half, RSE)
2	(half, RSE)
3	(half, RSE)
…	…

each block is a problem selected at random from the Patterson dataset

Approach 3

Block		Experimental units
1: NC_1, RS_1	a	(half, RSE)
	b	(half, RSE)
2: NC_2, RS_2	c	(half, RSE)
	d	(half, RSE)
…	…	…

each block represents a particular combination of NC and RS measures; a,b,c,… are problems selected at random from within each category of the Patterson dataset

The first approach is the simplest (cell means model). Problems are selected at random from the Patterson dataset and each treatment (buffer sizing technique) is applied. The random selection effectively treats the Patterson dataset as a population (which is itself a sample of the larger population of all possible network problems) from which a sample is drawn and examined, with conclusions applied not only to the sample but also to the total population. The model would be effective for noting the overal performance of one buffer sizing method with respect to another. However, this design alone will not facilitate additional conclusions about other factors expected to have an effect on the best technique (e.g., NC, RS, etc.). Actually, the model as shown is a modification to the basic cell means model that ensures each treatment is applied to each selected problem, effectively creating a repeated measures case. In the normal case, the cells in the table (where the a,b,c,… are shown) would have only one instance of each selected problem.

The second approach is a single factor experiment with repeated measures on all treatments. This is essentially the same as the first method except that it is more formally aligned with the standard terminology. The problems are viewed as blocks (usually called subjects as one way of demonstrating the repeated measures case) as a random sampling of the larger population. This model is more appropriate because it acknowledges that there are differences attributed to the blocks (i.e., that the best buffer sizing method may be problem-dependent).

The model is identical to the randomized block model except that it includes no replications. Hence, because there is no interference (as noted earlier) a combination of this model and the random block model is appropriate: it would allow for multiple replications (one iteration in the simulation for one problem for each of the buffer sizing techniques = one replication) and it would allow for repeated measures.

While the second approach is more formally correct than the first, it is similarly limited in the conclusions that can be drawn from it (i.e., overall, is one sizing method better than another?). While the approach would also note the significance of problem, it would not necessarily characterize the problem nor allow any inferences about how the problem structure or type affected the results.

The third approach acknowledges the problems can be categorized by their descriptive characteristics. It accommodates more than one blocking variable to allow 1) an overall assessment of whether one is better than another and 2) for combinations of blocking runs, which technique is best [is this true?]…

A 2^3 factorial design was ultimately chosen because it allowed inferences based on not only buffer sizing method but also the relative impact of other factors that were expected to influence the results. As noted earlier in this paper, it was expected that network characteristics such as NC and RS could have an impact on the buffer performance because highly complex nets and poor resource strength effectively create a highly dependent network. Such a strong dependence on other tasks creates greater likelihood of delay since an increase in dependence means more tasks are now available to delay any given task.

The 2^3 factorial design divides the factors into "high" and "low" levels. As noted by Montgomery (1997), although such a design does not characterize the effects of factors across their entire range, it works well for initial investigations (such as this one). The design is shown below. In it, the three main effects factors (A, B, and C) are chosen as buffer sizing method, net complexity, and average resource strength, respectively. Treatment labels are inserted in the body of the table. Two problems were assigned to each treatment cell and were run 500 iterations each.

Experimental Design with annotated treatments					
		B			
		NC (-)		**NC(+)**	
		C		C	
		Avg RSr (-)	**Avg RSr (+)**	**Avg RSr (-)**	**Avg RSr (+)**
A	**"Half"**	(1)	c	b	bc
	"RSE"	a	ac	ab	abc

Assumptions

The plan and corresponding simulation generally assumes independence of tasks. While there is dependence based on project task order logic and resource availability, no other significant interaction is assumed between the tasks. For example, the technical performance of one task does not affect the other tasks except from the point of view that the simulated duration somewhat accommodates the extended activity duration that would result from unforeseen technical difficulty. There is also no systemic effect of multi-task interaction that would have an overall common effect on the project. (An example of a systemic effect that could occur in real life is workforce illness: if bad food in the company cafeteria caused a number of workers to become ill, multiple apparently unrelated tasks could be expected to run long due to reduced resource availability.)

Limitations

Because the simulation is not a time-step simulation, one of the key aspects of critical chain management is removed from consideration. Specifically, one does not get to have insight into the ability of the different buffer sizing methods to provide adequate warning and support for management purposes. (Recall that in the CCPM concept, expected penetration of the buffers triggers management action to adjust project performance to help ensure on-time completion. It uses the threshold levels (i.e., at the one-third and two-thirds points) and simple rules (e.g., three consecutive reports that estimate increases in buffer consumption) as the trigger points.)

However, it also avoids the issue of developing a process for applying management action. Williams (1999) notes that identifying the actual specific effect of management action is not only difficult, it is often counter-intuitive. Developing a process for inclusion in this simulation would not only have potentially exceeded the scope of the effort, but also jeopardized the conclusions. After all, how would one determine which tasks to designate as adjustable (and to what extent), which precedence logic to designate as adaptable (perhaps with slight extension of some individual activities), etc. Accordingly, not including management action not only avoids a significant difficulty, it also removes a particularly difficult confounding effect that would be subject to attack.

Without management action, the simulation is essentially a project that is well-planned but that is executed on autopilot. Again, while this is not a perfect replication of reality it is nonetheless a reasonable representation. One should recognize that the buffers should be consumed in this autopilot mode. Indeed, as noted earlier, it could even be argued that right-sized buffers with no management action during the project execution will not only be fully consumed, but could even slightly overrun.

The sizing methods studied do not include Newbold's (1998) two standard deviation method nor Herroelen and Leus' (2001) modification to it. The method proposed by Hoel and Taylor (1999) is also not included. Newbold's method is not included due to the math error noted by Herroelen and Leus (2001) and Herroelen and Leus' method is not included for two reasons: it appears far less popular (Herroelen and Leus seem to offer it as a concession to the pre-existing Newbold method) and it requires the use of the lognormal distribution. Besides potential issues with normality assumptions necessary for the lognormal distribution, the available simulation tool did not conveniently handle the transform necessary to adapt the distribution for initial values greater than zero. Hoel and Taylor's method could not be examined here because it would have an unfair advantage due to the artificiality of the test environment.

Specifically, because the method to test buffer performance (Monte Carlo simulation of the network) would be the same as the one used to generate the values for Hoel and Taylor's method, it was recognized that it could not be included.

Only single mode, makespan minimization problems were examined. Accordingly, one must beware of generalizing conclusions beyond the range of the cases discussed in this paper.

Although uncertainty of the duration estimates was expected to have an effect, it could not be completed at this time. Accordingly, rather than allowing the variability of the d_j estimates to be random (i.e., when developing the "a" and "c" parameters for the triangular input distribution), the high variance case (a=b/3) was used with the expectation that the low variance case (a=2b/3) could be performed in a later work.

This section of the paper details the results and analysis of the experiment. It is divided into several sections: Filling the experimental design, Establishing the feeding buffers, Rounding

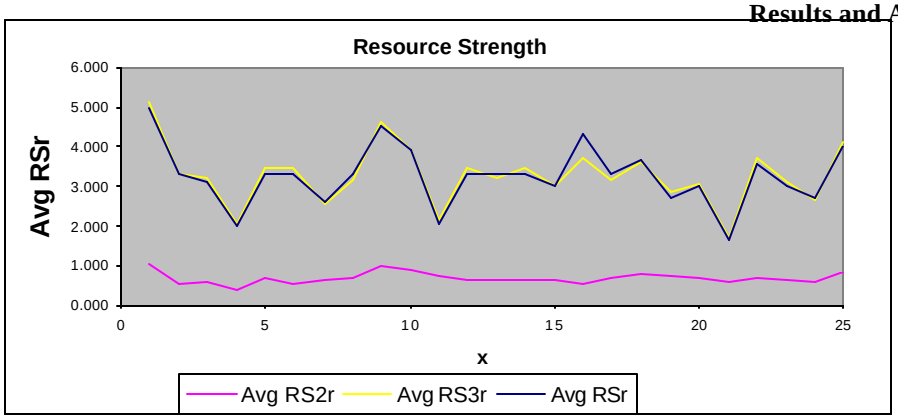
order seq #	Problem ID	# Activities J	# Resources R	Resource Max Availability Kr : r1	Kr : r2	Kr : r3	# of Arcs n-redunda	Network Complex NC	Resource Factor RF	Resource Strength Avg RSr	Avg RS2r	Avg RS3r
1	86	27	3	15	15	15	41	1.519	0.973	5.004	1.027	5.149
2	99	27	3	10	10	10	39	1.444	0.973	3.336	0.536	3.311
3	73	27	3	10	8	10	41	1.519	0.973	3.120	0.583	3.207
4	77	27	3	6	6	6	41	1.519	0.973	2.002	0.422	2.060
5	81	27	3	10	10	10	41	1.519	0.973	3.336	0.685	3.433
6	94	27	3	10	10	10	38	1.407	0.973	3.336	0.566	3.433
7	26	22	3	6	10	10	35	1.591	1.000	2.603	0.630	2.566
8	95	27	3	10	10	10	39	1.444	0.973	3.336	0.669	3.181
9	57	22	3	15	15	15	32	1.455	1.000	4.530	1.010	4.618
10	6	22	3	13	13	13	35	1.591	1.000	3.926	0.890	3.926
11	7	9	1	5	0	0	11	1.222	1.000	2.059	0.714	2.125
12	63	27	3	10	10	10	42	1.556	0.973	3.336	0.643	3.453
13	92	27	3	10	10	10	36	1.333	0.973	3.336	0.643	3.207
14	59	27	3	10	10	10	40	1.481	0.973	3.336	0.655	3.444
15	22	22	3	10	10	10	34	1.545	1.000	3.020	0.655	2.978
16	9	18	1	8	0	0	30	1.667	0.875	4.324	0.526	3.750
17	96	27	3	10	10	10	36	1.333	0.973	3.336	0.703	3.183
18	82	27	3	11	11	11	41	1.519	0.973	3.670	0.753	3.632
19	23	22	3	7	10	10	33	1.500	1.000	2.707	0.707	2.853
20	30	22	3	10	10	10	34	1.545	1.000	3.020	0.683	3.029
21	8	9	1	4	0	0	11	1.222	1.000	1.647	0.571	1.700
22	110	51	3	10	12	10	70	1.373	0.973	3.529	0.697	3.736
23	48	22	3	10	10	10	24	1.091	1.000	3.020	0.655	3.079
24	28	22	3	7	10	10	34	1.545	1.000	2.707	0.592	2.634
25	83	27	3	12	12	12	41	1.519	0.973	4.003	0.821	4.119

effects, Overview of the simulation results, and Scoring.

Filling the experimental design

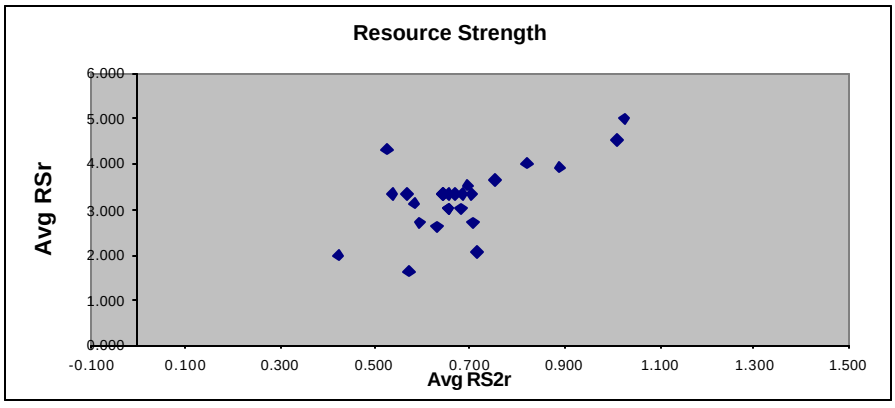
Network characteristic measures for a random selection of 25 problems from the Patterson dataset are in the table below. (A slightly expanded version of this table is included in Appendix D.)

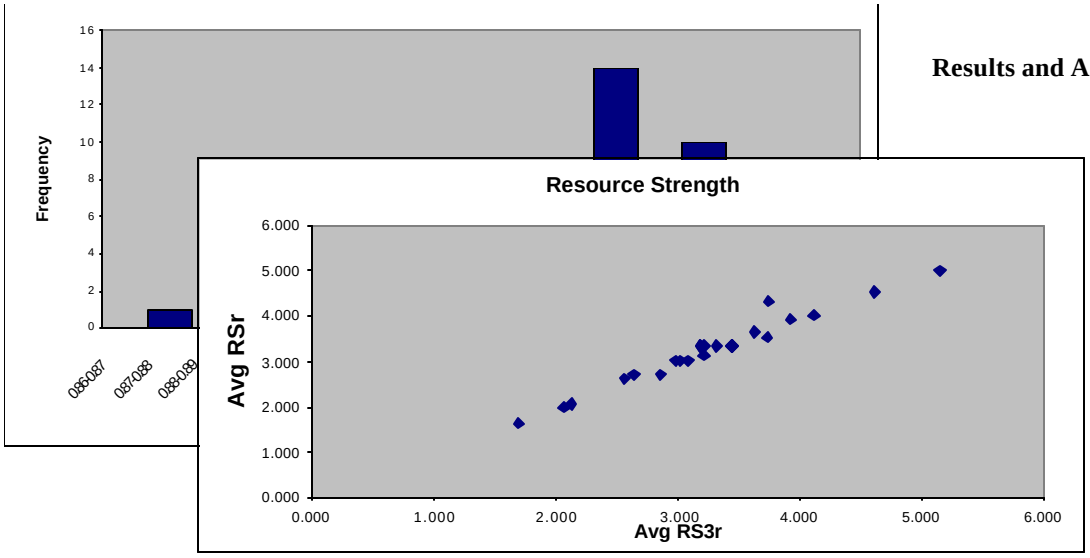
The figures below show plots of the values for the Resource Strength measures. In the first figure, all three Resource Strength measures are shown. Note that there is little difference between the average RSr and RS3r; it appears that Avg RSr is a slightly smoother line than the line for Avg RS3r. A more detailed examination of the data shows that, in general, Avg RS3r is slightly larger than Avg RSr. Because Avg RS2r varies across a tighter range than the other two, it is difficult to determine from the first figure whether it has the same relative shape as the other two. In the second figure, Avg RS2r is plotted separately and the scale is exploded to more clearly identify relative changes between adjacent values. Comparing the shape of this line to the lines for Avg RSr and Avg RS3r in the first figure, one can see that the general shapes of the three variables are the same. The most apparent difference occurs at observation 16: in the first figure, it corresponds to a relative high (adjacent values are lower) but in the second figure, it corresponds to a relative low (adjacent values are higher). Examining the data, one can see that observation 16 corresponds to problem 9, one of the few single-resource problems within the selection. There is also a difference at observation 11 (problem 7): in the first figure it corresponds to a relative low while in the second figure it seems to be “status quo”, with a less drastic change than expected. Inspecting the data, one can see that this observation also



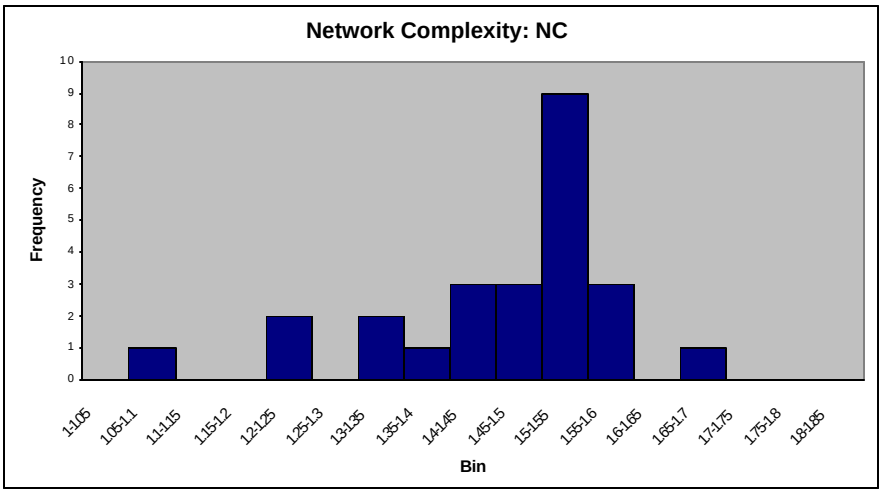
corresponds to another of the 3 problems in this random selection that have only one resource type. (The third problem does not exhibit this apparently anomalous behavior.)

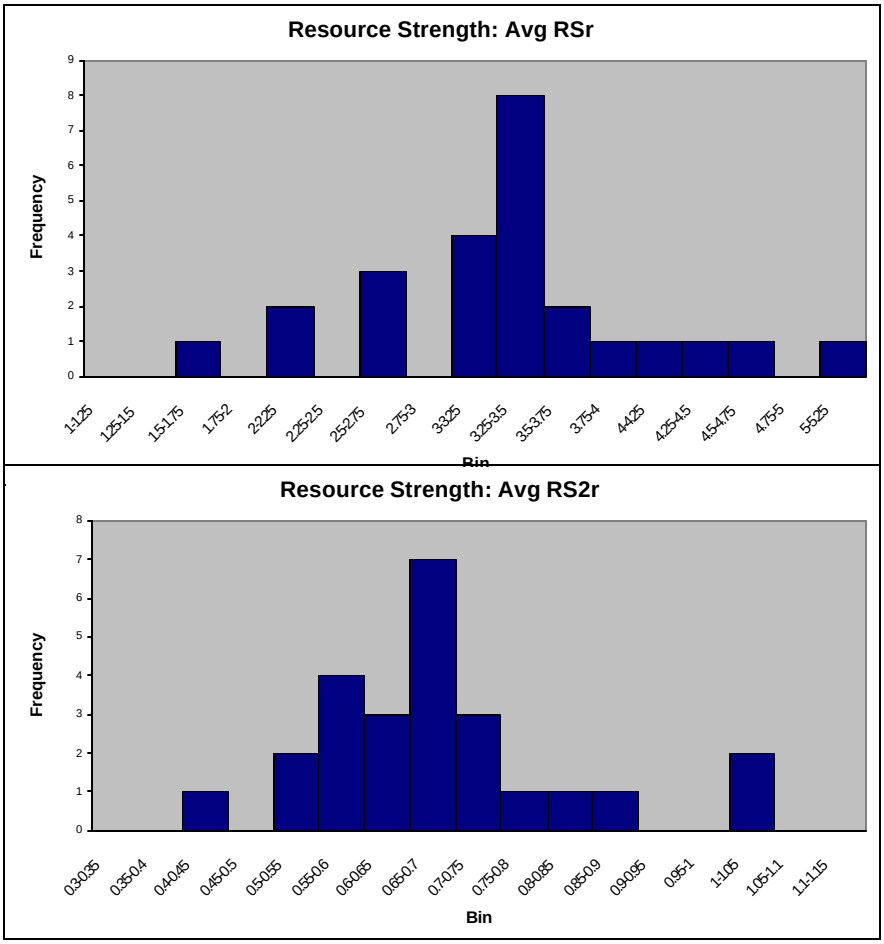
To confirm the correlation between the resource methods, values for Avg RS2r and Avg RS3r were plotted against Avg RSr. The results are shown in the two figures below. Note that there is a fair amount of variability between Avg RSr and Avg RS2r but very little between Avg RSr and Avg RSr3. Accordingly, subsequent investigations into resource strength as a measure would be better to investigate the impact of Avg RS2r rather than Avg RS3r.

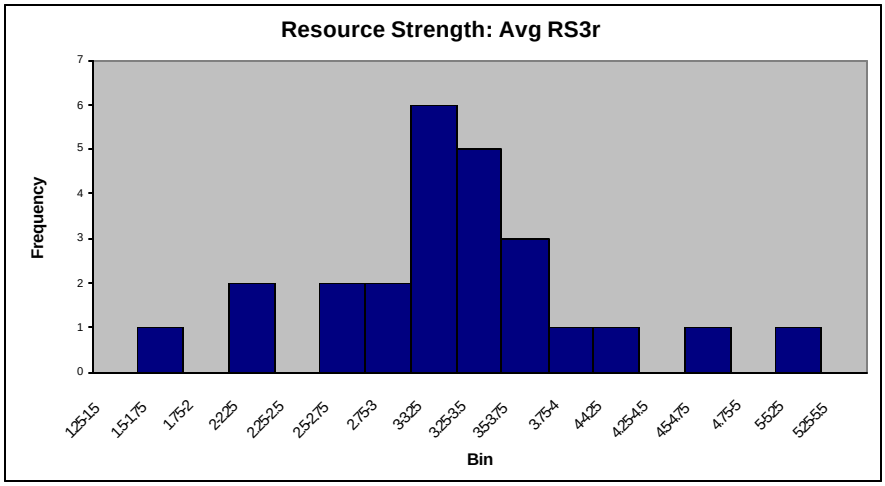




The figures below show histograms and ranges for each of the network characteristics for the same random selection of problems. (Data counts and summary statistics corresponding to these histograms are included in Appendix D, Section 1.) Note that there is a concentration of observations for each of the characteristics (i.e., they are not uniformly distributed throughout the range). Note further that the concentration occurs in the upper half of the range for NC but appears roughly in the middle of the range for the resource strength measures. The concentration is split between two values for resource factor (except for one observation at a lower value).







Because the values have a fairly strong central tendency, decomposing the range into bands to support a factorial experiment could prove problematic. Picking the dividing line at the center of the distribution can indeed yield half the observations above the value and half below it. However, the concentration of values would be near the adjacent ends of these range bands.

In other words, there might not be a significant enough difference to expect an influence by the factor. It would probably be best to force the "low" and "high" value selections to be separated from each other. One could do this by decomposing the range into three bands: a narrow central one and two larger ones at the ends. Then one could select "low" and "high" values from the larger bands.

After inspecting the histograms above, I had initially selected the range bands as noted below.

Factor	**Low band**	**High band**
NC	$1.05 \leq x \leq 1.45$	$1.5 \leq x \leq 1.7$
Avg RSr	$1.6 \leq x \leq 3.2$	$3.4 \leq x \leq 5.25$

This selection resulted in a reasonably even split: for the random selection of 25 problems, there were 9 and 13 problems in the low and high NC bands (respectively) and 10 and 7 in the low and high Avg RSr bands (respectively). However, I could not fill the experimental design (for a 2^3 factorial design) because there were an inadequate number of problems in each of the cells. The table below shows the number of problems that met the criteria above. (The first part of the table provides the counts for the random sampling of 25 of the problems; the second part of the table expands the sampling to 46 of the problems.) Note that nearly doubling the size of the sample did not result in any additional problems for the NC (-) $\cap$ Avg RSr (-) or NC (-) $\cap$ Avg RSr (+) cases. (Also, a number of problems in the sample failed to simultaneously meet NC and Avg RSr range band requirements. Accordingly, the number of problems within the table are less than the total sample size.)

Number of problems meeting criteria (sample n=25)		
	NC (-)	NC (+)
Avg RSr (-)	3	7
Avg RSr (+)	1	5

Number of problems meeting criteria (sample n=46)		
	NC (-)	NC (+)
Avg RSr (-)	3	13
Avg RSr (+)	1	8

Since there seemed to be no problem meeting the NC (+) $\cap$ Avg RSr (-) and NC (+) $\cap$ Avg RSr (+) cases, I considered adjusting the range bands in hopes of distributing the available problems to get more problems available for each cell. Inspecting the sample, I found that centering the narrow middle (exclusionary) band at Avg RSr = 3.3 was a good choice and that the 0.2 width of this band was also a good choice. This meant that the middle band was well-centered and that it was slightly wider than 1/4 standard deviation. Narrowing the band could result in inadequate differences between the "low" and "high" bands. Besides, the decomposition of Avg RSr values seemed to be less of a problem than the NC values.

Inspecting the data more closely shows that the initial estimation of the center of the NC distribution was slightly off. A better middle (exclusionary) band would be the interval (1.475, 1.525). For the random sampling of 46 problems, this would place 15 problems in the "low" band and 20 problems in the "high" band. The for this new range decomposition, the number of problems meeting the criteria are shown in the table below. Minor changes that tighten the

interval do not change the number of elements within each of the bands (until the middle range drops to near zero, an undesirable case).

Number of problems meeting criteria (sample n=25)		
	NC (-)	NC (+)
Avg RSr (-)	3	4
Avg RSr (+)	2	2

Number of problems meeting criteria (sample n=46)		
	NC (-)	NC (+)
Avg RSr (-)	7	10
Avg RSr (+)	2	7

Accordingly, the chosen intervals for NC and Avg RSr bands are shown in the table below.

Range intervals: NC, Avg RSr		
Factor	**Low band (-)**	**High band (+)**
NC	$1.05 \leq x \leq 1.475$	$1.525 \leq x \leq 1.7$
Avg RSr	$1.6 \leq x \leq 3.2$	$3.4 \leq x \leq 5.75$

Problems (ordered sequentially based on the previously-used random order) that apply to each cell of the design matrix are shown below. Note the dividing line between the initial 25-problem sample and where the sample was expanded to a 46-problem sample.

NC (-)		**NC (+)**		
Avg RSr (-)	**Avg RSr (+)**	**Avg RSr (-)**	**Avg RSr (+)**	
pat7	pat57	pat26	pat6	
pat8	pat110	pat22	pat9	n=25
pat48		pat30	pat15	
pat20		pat28	pat 14	
pat21		pat35	pat 5	
pat32		pat18	pat 4	
pat50		pat38	pat 104	n=46
		pat24		
		pat27		
		pat29		

Thus the layout of the experiment is as shown below.

	NC (-)		**NC (+)**	
	Avg RSr (-)	**Avg RSr (+)**	**Avg RSr (-)**	**Avg RSr (+)**
"Half"	pat7	pat57	pat26	pat6
	pat8	pat110	pat22	pat9
"RSE"	pat7	pat57	pat26	pat6
	pat8	pat110	pat22	pat9

It is interesting to note that of the eight problems, three were single-resource problems (pat7, 8 and 9) despite the fact that the selection process did not have an explicit bias related to the number of resources. The number of resources should not have an impact on buffer sizing or its effectiveness. The same constraining effect can occur regardless if there are one or many resources; the key difference is that with many resources there are simply more requirements to check. It is possible that an implicit bias (with respect to the number of resources) was present. In single resource problems, the Avg RSr is not tempered by additional resources. It is equivalent to Avg RSr for the single resource. For the multi-resource problems, the Avg RSr is the combination of the multiple resources, where high values in one resource *can* be balanced by low values in another. In hindsight, a more suitable measure might have been to select the minimum of the Avg RSr for each of the resources.

Network characteristics for NC and Avg RSr for the eight problems are consolidated in the table below. For additional characteristics, see the summary spreadsheet in Appendix D, Section 1. See Appendix D, Section 4 for large-format Gantt charts and network diagrams of the problem networks.

Problem ID	**NC**	**Avg RSr**
pat6	1.591	3.926
pat7	1.22	2.059
pat8	1.22	1.647
pat9	1.667	4.324
pat22	1.545	3.020
pat26	1.591	2.603
pat57	1.455	4.530
pat110	1.373	3.529

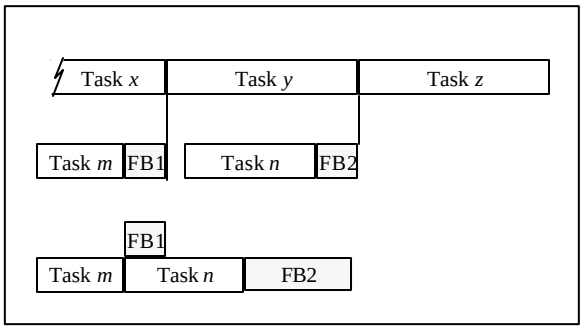
Establishing the Feeding buffers

An issue arises that is especially prevalent in highly-connected (i.e., relatively high NC) networks with weak resources. For a feeding buffer to be viable, there must be enough resources available for the preceding tasks to run into the buffer while still allowing the parallel tasks (especially, but not necessarily only the critical chain) to be executed. This should be (and perhaps is) inherent within the critical chain process. However, the concept is not discussed as a fine point in the literature, perhaps because of the simple examples commonly used. It may be part of the more detailed instruction provided in CCPM seminars.

The figure below shows two ways to plan the workload. The top line represents tasks on the critical chain. Below it are two options for scheduling tasks on a feeding chain. Assume that all tasks require the same resource type, that each task requires one unit of the resource and that there are two units of the resource available for the project throughout all time periods. (The result is that, during any one period, only one task can be accomplished simultaneously with the task on the critical chain.) Immediate successor tasks for task *m* includes task *y* and *n*; immediate successor for task *n* is task *z*.

In the first of options, each of the tasks on the feeding chain are separated by a FB. While this would guarantee viability re: no issue re: overrunning segments of the feeding chain, it appears counter to critical chain principles because it forces a pause in the schedule when one may not be necessary. (Consider the case where the first task on the feeding chain ends as scheduled. No feeding buffer is consumed and the subsequent task is held instead of beginning immediately.)

The second option shows a better case: the critical chain is protected by the FBs and each of the tasks on the feeding chain is allowed to run consecutively without pause. After all, the real use of the feeding buffers is to protect the critical chain.



However, an issue concerns the sizing of such FBs. Should they be based on the size of the feeding chain segment (i.e., those tasks since the start of the project or the last task that was a predecessor task of a task on the critical chain) or should they be based on the size of the larger cumulative segment of all tasks that are part of the same consecutive chain?

It seems more appropriate that for the buffers to be sized in accordance with the second (i.e., cumulative segments) method. The reasoning is best explained via example. Consider two tasks on a feeding chain and that the first is a long duration task while the second is short. Suppose that the first task consumes some but not all of its feeding buffer (i.e., overruns the planned end date for the task but does not delay the critical chain). The second task cannot begin until the preceding one is complete (due to the resource constraints in the preceding example). Even if this subsequent task performs in accordance with its estimated 50:50 time, it could overrun its segment buffer (which would be smaller than the preceding task's segment buffer).

In some cases, it is accepted to insert feeding buffers (or portions of them) onto the critical chain just prior to the project buffer. The method is used to prevent gaps in the critical chain that might otherwise occur when inserting feeding buffers at the end of a chain that includes a predecessor task on the critical chain. When inserting the buffer, the tasks on the feeding chain get pushed backward (earlier in time) until the buffer fits into the plan at the end of the feeding chain. However, if the buffer is large enough, it could push the predecessor tasks on the critical chain backward, creating a gap in the critical chain.

Leach (2000) describes the process to resolve the gaps, stating that if the gaps are small, one should insert the balance of the feeding buffer at the end of the project just prior to the project buffer. Although such a feeding buffer is immediately adjacent to the project buffer, considering the two as separate blocks allows them to be separately accounted for and helps prevent the buffer management reports from getting confusing. However, the literature does not define "small" nor does it indicate what one should do if multiple feeding buffers have this same problem on a single project.

There is also a danger of double counting, both within the plan as well as when running the simulation. This is especially true for cases where some portion of multiple feeding buffers on the same feeding chain are moved to just before the start of the project buffer (to preclude forcing gaps into the critical chain). Such buffers should be allowed to run simultaneously in order to preclude "double counting."

When establishing the critical chain plans for the problems in this report, I made no distinction for "small" versus "large." In other words, the balance of any feeding buffers that would otherwise create a gap in the critical chain was inserted on the critical chain just prior to the project buffer.

In a normal project where feeding buffers can be shown directly and completely at the end of their respective chains, overrunning the feeding buffers could create an additive delay of the project. Accordingly, I reasoned that the feeding buffers in this special case should also have the potential for an additive delay and I portrayed them consecutively prior to the project buffer. (For example, see the critical chain plan for problem pat22 in Appendix D, Section 2.) Again, this is a fine point that does not show up in the literature due to the typically simple problems that are used as examples.

Rounding effect

It was obvious that the practice of rounding up when converting "high confidence" estimates to their reduced values can be detrimental. It was expected that this would be the case due to the generally short activity times (all of the selected problems had single-digit durations in the original problem).

The effect is obvious in the problem patterson6, where each activity has a duration of 1 time unit. Following the established guidelines, reducing this "high confidence" estimate by 1/2 and then rounding up to the next integer value results in no change. Thus, the values in developing the critical chain plan are the same as the "high confidence" values. As expected, when running the simulation, buffer consumption did not exceed 0 because the upper value of each activity's duration distribution is the same as the plan. All of the simulated times fall below this value. Thus, such extreme cases make CCPM *worse* than reverting to traditional methods where safety time is distributed across each of the activities: not only does this same spread of safety time occur but an *additional* buffer is added.

One should thus beware when establishing a plan with low values for the "high confidence" estimates and should not blindly follow the guidelines. The intent of the "round up" concept was probably only to recognize that the estimating process is inexact and that a minor adjustment would be irrelevant. Also, displaying "half" periods portrays a precision that is not actually present. However, in reality, if time period estimates are low single-digit values, it is probable that the estimate has greater precision but needs a different scale (e.g., if all activities are estimated as "1 day" it would probably be better to convert to an "hours" schedule).

In hindsight, it might have been better to treat the activity durations in the original problems as the reduced times (50:50 times) instead of the "high confidence" times. Doing so would have prevented the effect of the "rounding." However, time available did not allow repeating the process with different activity times. One could not simply re-run the problems because the mix of activities in each problem (i.e., with and without rounding) would require new critical chain plans. However, the exception is patterson6, because *all* the activity durations are 1 time unit. Thus, the same adjustment could be applied to all activities and the chain logic would not need to be changed. Using "1" as the reduced times and changing the activity time distribution in the simulation would preclude re-creating a new critical chain plan and the corresponding simulation logic.

The original results for problem patterson6 had no iterations that consumed *any* project buffer. They were not included in the subsequent analysis. Instead, the activity time distributions were modified as described in the preceding paragraph (the original problem duration was considered the 50:50 duration). Accordingly, the original times were doubled to get new "b" values for the triangular distribution and the "a" and "c" values were then calculated in the same manner as for all the other problems. This problem is listed as patterson6-MODIFIED throughout this paper.

New a,b, and c values (= .67, 1.1, and 2, respectively) were inserted as the activity time distributions in the simulation and the simulation was repeated for both the "half" and "RSE" methods. Results of this modified problem were included in the analysis.

Overview of Simulation Results

Output data from the simulation runs is in Appendix E, Section 3. However, due to the magnitude of information, it consists of summary results only. More comprehensive data reports (e.g., listing the results of the feeding buffer consumption) are located within the enclosed electronic files. In general, the appendix is organized such that there are several pages of data for each problem. However, for convenience, a consolidated set of histograms for percent project buffer consumed (pb% consumed) is included at the beginning of the section.

The consolidated set allows a quick comparison between the two buffer sizing methods. Note that both the vertical and horizontal scales vary for each of the histograms. Nonetheless, one can see that, in general, the differences between the two methods for most of the problems appears very small. For example, for problem pat7, both methods had similar frequency of zero consumption percentages (approximately 23% and 33% for "half" and "RSE", respectively). One can also see that the balance of these percentages is scattered across a similar range (0-57% and 0-52%, respectively). There *are* differences but only a few of the problems appear to be particularly sensitive to the buffer sizing method.

Although it takes a close examination, one can see that the "RSE" method consumes more of the project buffer than the "half" method for pat26, including values up to 90% (compared to only 46% for the "half" method). A similar effect occurs for problem pat57 and

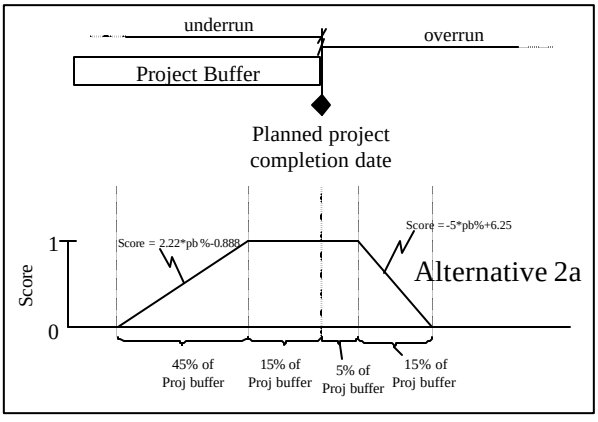
pat110. In pat110, only approximately 12% of the runs resulted in some non-zero amount of buffer consumption for the "half" method while the "RSE" method had a value closer to 70%.

Scoring

The scoring method that I had planned to use (previously referred to as "Alternative 2" (see Methodology section) could not be used. While I believe such thresholds are appropriate, leaving them in place would have resulted in little data to analyze. The table below shows counts for the number of non-zero scores for each of the problem-buffer sizing method combinations using Alternative 2 scoring. Recall that the simulation was run 500 iterations for each problem-buffer sizing method combination. (The identical values for both buffer sizing methods for problem patterson9 is due to two reasons. First, both methods resulted in equivalent project buffer sizes for this problem. Second, a common random number seed was used for the simulation.)

Count of non-zero scores based on scoring Alternative 2 (n=500)							
pat6MOD	pat6MOD	pat7	pat7	pat8	pat8	pat9	pat9
half	RSE	half	RSE	half	RSE	half	RSE
13	33	1	0	1	16	139	139
pat22	pat22	pat26	pat26	pat57	pat57	pat110	pat110
half	RSE	half	RSE	half	RSE	half	RSE
0	0	0	41	30	71	0	16

Accordingly, it was deemed necessary to adjust the scoring in order to allow more non-zero scores so that potential differences between the methods could be examined. The model is shown below and is referred to as "Alternative 2a". The only change from the planned model was the widening of the base of the trapezoid such that non-zero scores were produced for iterations that now had project buffer consumption percentages in the range $40\% \leq pb\% < 125\%$. The scoring model is depicted in the figure below.



Using the same simulation run data (i.e., the same percent project buffer consumed), scoring "Alternative 2a" was applied and the results were obtained. They are shown in the table below.

Count of non-zero scores based on scoring Alternative 2a (n=500)							
pat6MOD	pat6MOD	pat7	pat7	pat8	pat8	pat9	pat9
half	RSE	half	RSE	half	RSE	half	RSE
73	129	41	29	44	84	290	290
pat22	pat22	pat26	pat26	pat57	pat57	pat110	pat110
half	RSE	half	RSE	half	RSE	half	RSE
0	1	16	122	184	254	0	89

Note that there is generally a large increase in the number of non-zero scores with scoring Alternative 2a. This implies that a fair number of projects *were* consuming their buffers but only at relatively low amounts. That is, both buffer sizing methods seem to overestimate the necessary safety time. (See Appendix E, Section 3.) One can confirm from the histograms for the project buffer consumed (pb consumed) that projects were indeed consuming some amount of buffer. (Remember that "pb consumed" is in time units while "pb % consumed" is a percentage.)

Few of the runs for either buffer method achieved a score of 1. Counts are provided in the table below.

Count of scours = 1 based on scoring Alternative 2a (n=500)							
pat6MOD	pat6MOD	pat7	pat7	pat8	pat8	pat9	pat9
half	RSE	half	RSE	half	RSE	half	RSE
4	7	0	0	0	1	49	49
pat22	pat22	pat26	pat26	pat57	pat57	pat110	pat110
half	RSE	half	RSE	half	RSE	half	RSE
0	0	0	9	3	11	0	4

One of the differences noted between the methods is that, compared to the "half" method, the "RSE" method essentially shifts safety time from the project buffer to the feeding buffers. This is due to the effect that for short chains the "RSE" method results in longer buffers than the half method, with just the opposite effect on longer chains. Accordingly, project buffers, because they are developed from the relatively long critical chain, are usually shorter in "RSE" than via the "half" method; feeding buffers are usually longer. However, the feeding chains in these problems are potentially unnaturally short due to the high interconnectivity of the feeding and critical chains. Given this complementary effect of the "RSE" buffer sizes for long critical chains and short feeding chains, there is a potential balance point where both methods provide a similar amount of safety time. To confirm whether this phenomenon was occurring, I examined the feeding and project buffer sizes for the two methods across all the selected problems. The results are in the table below. Note that the "RSE" method had a slightly larger average feeding buffer size but that its project buffer size was dramatically shorter. If the feeding and critical chains were less inter-connected (i.e., the feeding chains were allowed to be longer due to relative independence), the feeding chains would be longer than in these problems and then both the project and feeding buffers would be smaller via the "RSE" method, thereby reducing overall safety time.

Problem ID	"Half" Method		"RSE" Method	
	Avg FB size	PB size	Avg FB size	PB size
pat6MOD	1	4	1	2
pat7	1	3	1.67	3
pat 8	1	4	1.5	3
pat 9	2.4	6	2.8	6
pat 22	3	12	3.29	7
pat 26	1.2	14	1.8	7
pat 57	2.75	6	2.25	5
pat 110	6.31	27	6.31	13
Overall Average	2.3	9.5	2.6	5.8

It is further possible that the true effect of the sizing methods were masked due to the high interconnectivity between feeding and critical chains. As noted earlier, a number of feeding buffers were inserted on the critical chain prior to the project buffer to preclude gaps in the critical chain. While the practice is acceptable, it can hide the result of the project buffer consumption because the feeding buffer (if not consumed by the feeding chain) is now available to be consumed by the critical chain.

Inserting the score counts for scoring Alternative 2a into the experiment matrix seems to indicate that the "RSE" method is superior for all combinations of NC and RSr. Not only does it achieve better scores than the "half" method for every combination, it also is better for almost every problem. (Note that both are equivalent at 290 for pat9 and that they are virtually equivalent at 0 and 1 for pat22.)

Count of non-zero scores based on Scoring Alternative 2a (n=500 each cell)				
	NC (-)		**NC (+)**	
	Avg RSr (-)	**Avg RSr (+)**	**Avg RSr (-)**	**Avg RSr (+)**
"Half"	41 44	184 0	16 0	73 290
"RSE"	29 84	254 89	122 1	129 290

Repeating the table with the average score results in the following table. Recall that the score range is [0,1].

Average score based on Scoring Alternative 2a (n=500 each cell)				
	NC (-)		**NC (+)**	
	Avg RSr (-)	**Avg RSr (+)**	**Avg RSr (-)**	**Avg RSr (+)**
"Half"	.015 .016	.111 0	.004 0	.044 .314
"RSE"	.009 .058	.203 .056	.109 0	.098 .314

While the results are not what I would call desirable, it is encouraging that there are positive values. This indicates that buffers are sized such that projects are entering the region where there is neither too little nor too much safety time.

In general, the "RSE" method appears better across all the combinations. Indeed, only for problem pat7 does "half" score better than "RSE" (scores are .015 and .009, respectively). Scores are equivalent for two of the problems (pat9 and 22 with scores of .314 and 0, respectively). Although the percent increase of one method over another may be sizeable, the raw scores are nowhere near as dramatic as I had expected.

To conduct a more detailed analysis of variance, the average scores in the preceding table were replaced by score sums and the table was extended to include row and column totals. The table is shown below. Because of its size, the raw score data is not included in an appendix but the data is available on the enclosed electronic media.

Score sums by treatment combination, scoring alternative 2a (n=1000 each cell)

	NC(-)		NC (+)		
	RSr (-)	**RSr (+)**	**RSr (-)**	**RSr (+)**	Row Total
"Half"	15.4386	55.46324	2.078031	179.2239	252.2037
"RSE"	33.163	129.6802	54.73699	206.265	423.8452
Col Total	48.6016	185.1434	56.81502	385.4888	676.0489

Both problems in each cell were combined (i.e., scores were added for each of the 1000 observations across the two problems in each cell). It is possible that some type of blocking based on the problem could improve the quality of these results. The resulting analysis of variance table is shown below.

ANOVA Table

Source	**SS**	**DF**	**MS**	**F**	**P-value**
Buffer method (A)	3.682598	1	3.682598	87.50416	1.07258E-20
Net complexity (B)	5.437097	1	5.437097	129.1937	1.03936E-29
Resource strength (C)	27.0532	1	27.0532	642.8253	1.8033E-136
AB	0.018731	1	0.018731	0.445082	0.504699061
AC	0.119156	1	0.119156	2.831325	0.092481103
BC	4.614336	1	4.614336	109.6437	1.71411E-25
ABC	0.842765	1	0.842765	20.02539	7.74791E-06
Error	336.342	7992	0.042085		
Total	378.1099	7999			

From the table, one can see that the mean square for all treatments is larger than the mean square for error. Thus, all treatments have an effect with the largest effects by resource strength, net complexity. Note that the effects due to buffer sizing method rank fourth behind the net complexity-resource strength interaction. Nonetheless, the effects are significant as confirmed by the P-value less than .0001.

I suspect that if the previously-described adaptation for Avg RSr were used instead Avg RSr, its effect would be more significant than that of Avg RSr above. This measure, like network complexity, has the potential to slow network performance due to delays on a parallel but connected path.

Both buffer sizing methods scored much more poorly than I had expected. While it is possible that some of the result can be attributed to potential masking of project buffer consumption due to forced placement of feeding buffers on the critical chain, it is unlikely because only a few of the networks had critical chain plans with such a structure.

Most problems had networks with short chains. This is perhaps due to the fact that they were highly resource constrained problems and that they shared the same (generally 3) resources. This meant that resource comparison (linking tasks due to resource limitations) was just as likely to drive a particular sequence as precedence logic.

These short chains frequently meant that the strings of tasks could not be "chains" at all; while there may have been a logic sequence, tight resource constraints prevented flexibility on parallel chains as parallel tasks were forced onto a task-for-task schedule comparable to the critical chain. The net effect was that, in a number of cases, chains were only 1 to 2 time periods long.

Both buffer sizing methods seem to overestimate the necessary safety time. This may be less of an issue with longer, less inter-connected chains. In such cases, the "RSE" method will result in lower buffer size estimates and likely result in higher scores. It is, however, difficult to determine without investigation whether the "RSE" would then underestimate the buffer and result in scores of zero due to overruns.

Depending on the length of the feeding chains, the "RSE" method may simply shift safety time from the project buffer to the feeding buffers. However, for long chains, the "RSE" method will result in less overall safety time (both in feeding and project buffers). Accordingly, one may choose one buffer sizing method over another depending on the length of the chains involved in the project, the expected risk level, and the uncertainty of the estimates.

While the buffer sizing method did explain a significant amount of the variation of the scores, it was surpassed by the network complexity, the resource strength, and the interaction between the two as a source of variation. Of the two methods, the "RSE" method performed better than the "half" method.

The additional resource strength measures did not make as significant a difference as I had expected. For future experiments involving network characterization, Avg RS2r may hold more promise than Avg RS3r.

Some fine points of critical chain scheduling are not adequately discussed in the critical chain literature, perhaps due to the relatively simple problems used as examples. While texts such as Leach (2000) and Newbold (1998) make great strides in explaining CCPM to the practitioner, holes still exist. Particular areas include resolving gaps in the critical chain and sizing feeding buffers on chains that have multiple inter-connections with the critical chain.

There are a number of areas for additional research or areas in which to improve this report.

The most noteworthy concerns the problem selection. There are two main parts to this issue:

1) Reconsider whether the Patterson dataset is adequately representative of projects or at least characterize its primary differences. Recognize that the problem set was developed to test solution methods for the resource constrained project scheduling problem. Accordingly, the set may be characterized as skewed toward a level of difficulty not necessarily common in practice. An initial step would be to expand the dataset summary sheet for the network characteristics (see Appendix D, Section 1).

2) Expand the problems considered to a larger set, either within the Patterson dataset or via others. The Project Scheduling Problem Library (see Appendix B) contains a large number of other problems. Although most of them are multi-mode problems, there are plenty of single-mode problems. It would be beneficial to compare them to the Patterson dataset.

Another obvious area for additional research, is to investigate the effects of other characteristics or to refine those used in this paper. For example, a potentially more appropriate measure of the resource difficulty is the minimum of the resource strength (possible either as a minimum of the average resource strengths by resource or as a minimum of the individual k_{jr}). Task estimate uncertainty is another likely network characteristic that could have a correlation to the preferred buffer sizing method. Since this paper used a "high uncertainty" case rather than a random uncertainty level, one could investigate the effect of uncertainty simply by expanding this investigation to include a "low uncertainty" case.

When investigating the effects of other characteristics, consider that there may be a non-linear response for resource strength. In a network with exceedingly strong resources, task precedence logic drives the order of tasks. For the same network structure but with weaker resources, determining the path through the network can be difficult because one is trying to work at maximum capacity. At this maximum capacity case, additional inter-relations are created. If the same net had extremely weak but (feasible resources), the resource constraints play a very significant role, at the extreme, converting the initially complex network into an entirely serial structure. Identifying whether different buffer methods would be preferred in these differing instances of the same network would be interesting and potentially beneficial work.

Future expansion of this paper should confirm nuances of critical chain accepted practice and ensure that these nuances are accommodated within the critical chain plan. Primary among these is the concept of feeding buffers on the critical chain prior to the project buffer. Also included is the concept of a feeding chain with multiple links to the critical chain: should these segments be considered as separate chains or as segments of the larger chain? Another point concerns whether it is appropriate to add buffers into chains that merge into a feeding chain and that subsequently merge into the critical chain. Should such buffers exist, should they be sized differently? Another obvious nuance concerns the concept of rounding the durations to integer values. While such a guideline is certainly common in the literature, it is clearly not appropriate for all cases (see the earlier discussion about problem #6).

Obviously, since one of the limitations of this paper was that it did not investigate all the popular buffer sizing methods, an area for future research would be to expand the experiment to investigate their relative performance. While using the lognormal distribution could allow the

addition of Herroelen and Leus' (2001) modified 2-Standard Deviation method, it appears difficult to develop a fair test of Hoel and Taylor's (1999) method.

It may be a good idea to change scoring method to the Alternative 3 described in the Methodology section of this paper. Doing so would allow non-zero scores to be developed, even though they may be at very low levels. Since many of the runs in this initial investigation resulted in low project buffer consumption, expanding the scoring range might help to differentiate between the methods.

Alternatively, one could simply adopt scoring Alternative 2 and widen the base of the trapezoid to encompass the entire project buffer in the left tail; extending the he right tail may or may not be necessary.

Another recommendation for future work would be to attempt to identify a better buffer sizing method or at least a few guiding principles about adapting the existing methods. That is, one could provide adjustment coefficients to the base methods that would be related to the network characteristics.

Some other lessons learned could help future researchers.

Consider using a supporting tool (e.g., ProChain to develop critical chain plans). While I had avoided using it because I wanted to test the concepts of the buffer sizing methods instead of validating the software, it could still be beneficial in establishing a recommended plan that could then be individually confirmed and adapted if necessary. The @RISK tool to run the Monte Carlo simulations via Microsoft Project could also be beneficial, potentially reducing the need to re-create problem structures in multiple programs.

I would recommend using the original problem durations as the "reduced" times to eliminate the effects of rounding to integer values. Although rounding would still apply to the buffer size, at least it would not necessarily be rampant throughout the network.

Sources:

Badiru AB. A simulation approach to PERT network analysis. Simulation 1991; 57(4): 245-255

Brucker P, Drexl A, Mohring R, Neumann K, Pesch E. Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research 1999; 112(1): 3-41

Cook DP. A simulation comparison of traditional, JIT, and TOC manufacturing systems in a flow shop with bottlenecks. Production and Inventory Management Journal 1994; 35(1): 73-78

Cooper DF. Heuristics for scheduling resource-constrained projects: An experimental investigation. Management Science 1976; 22: 1186-1194

Elmaghraby, Salah E. On criticality and sensitivity in activity networks. European Journal of Operational Research 2000; 127: 220-238

Goldratt EM. Critical chain. Great Barrington (MA): North River Press; 1997. 246 p.

Grey S. Practical Risk Assessment for Project Management. Chichester, England: John Wiley and Sons Ltd, 1995. 140 pp

Gutierrez GJ, Kouvelis P. Parkinson's Law and its implications for project management. Management Science 1991; 37(8): 990-1001

Herroelen W, Leus R. On the merits and pitfalls of critical chain scheduling. Journal of Operations Management 2001; 19(5): 559-577

Herroelen W, Leus R, Demeulemeester E. Critical chain project scheduling: Do not oversimplify. Unpublished article 2001;

Hoel K, Taylor SG. Quantifying buffers for project schedules. Production and Inventory Management Journal 1999; 40(2): 43-47

Kamburowski J. New validations of PERT times. Omega 1997; 25(3): 323-328

Kolisch, R, Sprecher A. PSPLIB - A project scheduling problem library. European Journal of Operational Research 1997; 96(1): 205-216

Kolisch R, Sprecher A, Drexl A. Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science 1995; 41(10): 1693-1703

Leach LP. Critical chain project management improves project performance. Project Management Journal 1999; 30(2): 39-51

Leach LP. Critical chain project management. Boston: Artech House; 2000. 330 p.

Montgomery DC. Design and Analysis of Experiments (4th ed). New York: John Wiley & Sons, Inc.; 1997. 704 pp

McKay KN, Morton TE. Critical chain. IIE Transactions Aug 1998; 30(8): 759-762

Neter J, Wasserman W, Kutner MH. Applied Linear Statistical Models (3d edition). Boston: Richard D. Irwin, Inc.; 1990. 1181 pp

Newbold RC. Project management in the fast lane: Applying the theory of constraints. Boca Raton: St. Lucie Press; 1998. 284 pp

Pascoe TL. Allocation of resources. C.P.M. Revue Francaise Recherche Operationelle 1966; 38: 31-38

Patterson J. A comparison of exact procedures for solving the multiple constrained resource project scheduling problem. Management Science 1984; 30(7): 854-867

Ramsay ML, Brown S, Tabibzadeh K. Push, pull and squeeze shop floor control with computer simulation. Industrial Engineering 1990; 22(2): 39-45

Steyn H. An investigation into the fundamentals of critical chain project scheduling. International Journal of Project Management 2000; 19(6): 363-369

Wendling RV, Lorance RB. Basic techniques for analyzing and presenting schedule risk analysis. AACE Transactions 1999: 8.1 - 8.8

Williams T. Towards realism in network simulation. OMEGA International Journal of Mgmt Science 1999; 27(3): 305-314